Presidenza del Consiglio dei Ministri

Dipartimento per la trasformazione digitale

DECRETO

Adozione delle Specifiche Tecniche del Sistema IT-Wallet durante la Sperimentazione, ai sensi dell'articolo 6, comma 1, del decreto del Presidente del Consiglio dei ministri 17 giugno 2026, adottato in attuazione dell'articolo 64-quater, comma 3, del decreto legislativo 7 marzo 2005, n. 82.

IL CAPO DEL DIPARTIMENTO PER LA TRASFORMAZIONE DIGITALE

Visto il decreto legislativo 7 marzo 2005, n. 82, recante il Codice dell'amministrazione digitale e, in particolare, l'articolo 64-quater, che disciplina il Sistema IT-Wallet;

Visto il Regolamento (UE) n. 910/2014 del Parlamento europeo e del Consiglio, del 23 luglio 2014, come modificato dal Regolamento (UE) 2024/1183 del Parlamento europeo e del Consiglio, dell'11 aprile 2024, relativo all'istituzione del quadro europeo relativo a una identità digitale europea;

Visto il decreto del Presidente del Consiglio dei ministri del 17 giugno 2026, adottato ai sensi dell'articolo 64-quater, comma 3, del decreto legislativo 7 marzo 2005, n. 82, recante l'approvazione delle Linee Guida del Sistema IT-Wallet e delle regole per la Sperimentazione;

Visto, in particolare, l'articolo 6, comma 1, del predetto decreto del Presidente del Consiglio dei ministri del 17 giugno 2026, il quale prevede che, durante la Sperimentazione, le Specifiche Tecniche, predisposte e aggiornate in modalità aperta e collaborativa avvalendosi degli strumenti messi a disposizione dal Dipartimento per la trasformazione digitale, sono adottate con decreto del Capo del Dipartimento per la trasformazione digitale, pubblicate sul Portale e contestualmente comunicate all'AgID e all'Agenzia per la cybersicurezza nazionale, sentite per gli aspetti di competenza;

Visto il decreto del Presidente del Consiglio dei ministri del 19 marzo 2026, adottato ai sensi dell'articolo 64-quater, comma 5, del decreto legislativo 7 marzo 2005, n. 82, recante la disciplina del Sistema IT-Wallet e dei relativi servizi;

Visto il Regolamento (UE) 2016/679 del Parlamento europeo e del Consiglio, del 27 aprile 2016, relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali, nonché alla libera circolazione di tali dati;

Visto il decreto legislativo 30 giugno 2003, n. 196, recante il Codice in materia di protezione dei dati personali;

Visto il decreto del Segretario generale del 24 luglio 2019, con cui è disciplinata l'organizzazione interna del Dipartimento per la trasformazione digitale, modificato dal decreto del Ministro per l'innovazione tecnologica e la digitalizzazione 3 settembre 2020, registrato dalla Corte dei conti in data 21 settembre 2020 al n. 2159;

Presidenza del Consiglio dei Ministri

Dipartimento per la trasformazione digitale

Visto il decreto del Presidente della Repubblica del 31 ottobre 2022, con il quale il Sen. Alessio Butti è stato nominato Sottosegretario di Stato alla Presidenza del Consiglio dei ministri;

Visto il decreto del Presidente del Consiglio dei ministri del 25 novembre 2022, concernente la delega di funzioni in materia di innovazione tecnologica e transizione digitale al Sottosegretario di Stato alla Presidenza del Consiglio dei ministri, Sen. Alessio Butti;

Visto il decreto del Presidente del Consiglio dei ministri 24 novembre 2022, con il quale al Dott. Angelo Borrelli è stato conferito, ai sensi degli articoli 18 e 28 della legge 23 agosto 1988, n. 400, nonché dell'articolo 19 del decreto legislativo 30 marzo 2001, n. 165, l'incarico di Capo del Dipartimento per la trasformazione digitale;

Considerato che le Specifiche Tecniche del Sistema IT-Wallet definiscono l'architettura tecnica, il quadro di implementazione e i requisiti di progettazione delle Soluzioni Tecniche del Sistema IT-Wallet;

Considerato che le Specifiche Tecniche sono state predisposte e aggiornate in modalità aperta e collaborativa, avvalendosi degli strumenti messi a disposizione dal Dipartimento per la trasformazione digitale;

Considerato che l'adozione delle Specifiche Tecniche durante la Sperimentazione è funzionale ad assicurare l'attuazione uniforme, interoperabile e sicura del Sistema IT-Wallet, nel rispetto delle Linee Guida del Sistema IT-Wallet e delle regole per la Sperimentazione;

Ritenuto di dover procedere, ai sensi dell'articolo 6, comma 1, del decreto del Presidente del Consiglio dei ministri del 17 giugno 2026, all'adozione delle Specifiche Tecniche del Sistema IT-Wallet durante la Sperimentazione;

Sentita l'Agenzia per la cybersicurezza nazionale che si è espressa con nota del 15/09/2026 acquisita a protocollo DTD-0005917-A-15/09/2026;

Sentita l'Agenzia per l'Italia Digitale che si è espressa con nota del 18/09/2026 acquisita a protocollo DTD-0006060-A-21/09/2026;

DECRETA

Art. 1
(Oggetto)

1. Il presente decreto adotta, ai sensi dell'articolo 6, comma 1, del decreto del Presidente del Consiglio dei ministri del 17 giugno 2026, le Specifiche Tecniche del Sistema IT-Wallet di cui all'Allegato 1, che costituisce parte integrante e sostanziale del presente decreto.

2. Le Specifiche Tecniche, di cui all'Allegato 1, sono in lingua inglese con relativa traduzione non asseverata in lingua italiana, ai sensi dell'articolo 6, comma 1, del decreto del Presidente del Consiglio dei ministri del 17 giugno 2026.

3. Le Specifiche Tecniche si applicano durante la Sperimentazione secondo quanto previsto dal decreto del Presidente del Consiglio dei ministri del 17 giugno 2026 e dagli ulteriori atti attuativi del Sistema IT-Wallet.

Art. 2
(Pubblicazione, entrata in vigore e comunicazioni)

1. Il presente decreto è pubblicato sul sito internet www.governo.it. Il presente decreto entra in vigore il giorno successivo alla pubblicazione.

2. Della pubblicazione del presente decreto è dato avviso nella Gazzetta Ufficiale della Repubblica Italiana.

IL CAPO DEL DIPARTIMENTO

Angelo Borrelli

24-09-2026 | 18:08:16 CEST

IT-Wallet Technical Specifications

Version 1.4.7

Release date: 2026-09-22

TABLE OF CONTENTS

1 Introduction **3**

1.1 Scope . . . 4

1.2 Normative Language and Conventions . . . 5

1.3 How to Read the Specification . . . 6

1.3.1 Specification Structure Overview . . . 6

1.3.2 Reading Paths by Role . . . 7

2 Brand Identity **15**

2.1 Naming . . . 15

2.2 Visual Identity . . . 16

2.2.1 Logo . . . 16

2.2.2 Trust Mark . . . 17

3 Architecture Overview **19**

4 Onboarding System **23**

4.1 Onboarding System Architecture . . . 23

4.2 Authentic Source Registration Process . . . 25

4.3 Federation Onboarding Process . . . 26

4.3.1 Onboarding of Relying Party Intermediaries . . . 26

4.4 Entity Lifecycle Management . . . 27

4.4.1 Ongoing Operations Management . . . 27

4.4.2 Federation Exit and Removal Processes . . . 28

4.4.3 Lifecycle Coordination Requirements . . . 29

4.5 Onboarding Journey Maps . . . 29

4.5.1 Ecosystem Overview . . . 30

4.5.2 Authentic Source Operator Journey . . . 31

4.5.3 Credential Issuer Operator Journey . . . 32

4.5.4 Wallet Provider Operator Journey . . . 33

4.5.5 Relying Party Operator Journey . . . 33

4.6 User Experience Journey . . . 34

5 User Experience Design **35**
5.1 Design Principles . . . 35
5.2 Functionalities Overview . . . 36
5.3 Activation of the Wallet Instance . . . 37
5.3.1 Focus on Person Identification Attestations . . . 38
5.4 Issuance of Electronic Attestations of Attributes . . . 42
5.4.1 Issuance from the Wallet Instance Catalog . . . 43
5.4.2 Issuance from a Touchpoint of the Authentic Source . . . 45
5.4.3 Focus on Electronic Attestations of Attributes . . . 55
5.5 Presentation of Electronic Attestations . . . 62
5.5.1 Proximity Presentation . . . 64
5.5.2 Remote Presentation . . . 65
5.6 Management of Electronic Attestations . . . 94
5.6.1 Status of Electronic Attestations . . . 94
5.6.2 Transaction Logging . . . 100
5.6.3 Backup and Restore of Electronic Attestation of Attributes . . . 101
5.7 Deactivation of the Wallet Instance . . . 101
5.8 Error Management . . . 103
5.8.1 Activation of the Wallet Instance Errors . . . 104
5.8.2 Issuance of Electronic Attestations of Attributes Errors . . . 104
5.8.3 Presentation of Electronic Attestations Errors . . . 105
5.8.4 Management of Electronic Attestations Errors . . . 105
5.8.5 Deactivation of the Wallet Instance Errors . . . 105
5.9 User Assistance . . . 105
5.10 User Feedback . . . 107

6 The Infrastructure of Trust **109**
6.1 Federation Roles . . . 110
6.2 Trust Infrastructure and Registry Integration . . . 111
6.3 General Properties . . . 111
6.4 Trust Infrastructure Requirements . . . 112
6.5 Federation API endpoints . . . 114
6.6 Configuration of the Federation . . . 116
6.7 Entity Configuration . . . 117
6.7.1 Entity Configurations Common Parameters . . . 118
6.7.2 Entity Configuration Trust Anchor . . . 119
6.7.3 Trust Marks for Relying Party Intermediaries . . . 119
6.7.4 Entity Configuration Leaves and Intermediates . . . 120
6.7.5 Metadata Types . . . 121
6.8 Metadata of federation_entity Leaves . . . 122
6.9 Subordinate Statements . . . 123

6.9.1 Subordinate Statement . . . 125
6.10 Federation Discovery . . . 125
6.11 Trust Chain . . . 127
6.11.1 Offline Trust Attestation Mechanisms . . . 130
6.11.2 Trust Chain Fast Renewal . . . 130
6.12 Trust Evaluation Mechanism . . . 131
6.12.1 Establishing Trust with Credential Issuers . . . 132
6.12.2 Establishing Trust with Relying Party . . . 133
6.12.3 Evaluating Trust with Wallets . . . 137
6.13 Non-repudiability of the Long Lived Attestations . . . 137
6.14 X.509 PKI . . . 138
6.14.1 Federation Trust Anchor and X.509 CA . . . 139
6.14.2 X.509 Certificates Issuance . . . 139
6.14.3 X.509 Certificate Revocation . . . 145
6.14.4 Federation Subordinate Events Endpoint . . . 146
6.15 Privacy Remarks . . . 147
6.16 Considerations about Decentralization . . . 148

7 Registry Infrastructure 149
7.1 Registry Architecture Overview . . . 149
7.2 Registry Discovery Endpoint . . . 150
7.2.1 Registry Discovery Endpoint Parameters . . . 150
7.3 Claims Registry . . . 152
7.3.1 Claims Registry Usage . . . 153
7.3.2 Claims Registry Structure . . . 154
7.4 Authentic Source Registry . . . 163
7.4.1 Authentic Source Registry Usage . . . 164
7.4.2 Public vs Private AS Coordination . . . 164
7.4.3 AS Registry Structure . . . 165
7.4.4 AS-CI Coordination . . . 178
7.5 Federation Registry . . . 178
7.5.1 Registry Integration Role . . . 178
7.5.2 Federation Registry Access . . . 179
7.6 Digital Credentials Catalog . . . 179
7.6.1 Digital Credentials Hierarchy . . . 183
7.6.2 Digital Credentials Catalog Structure . . . 194
7.6.3 Decentralization of Display and Claim Information . . . 203
7.7 Taxonomy . . . 203
7.8 Schema Registry . . . 219
7.9 Registry Integration and Cross-References . . . 221
7.10 Registry Infrastructure Usage Journeys . . . 222

7.10.1 Catalog Browsing . . . 222
7.10.2 Credential Issuance . . . 223
7.10.3 Credential Presentation and Verification . . . 223

8 Entity Onboarding 225
8.1 Overview . . . 225
8.1.1 Entity Onboarding System Architecture . . . 225
8.1.2 Entity Types and Onboarding Pathways . . . 225
8.1.3 Administrative and Technical Registration . . . 226
8.2 Authentic Sources Registration Process . . . 227
8.2.1 Authentic Source Registration Requirements . . . 227
8.2.2 Authentic Sources Registration Information Requirements . . . 228
8.2.3 Authentic Source Registration Procedure . . . 230
8.3 Authentic Source to Credential Issuer Authorization Process . . . 240
8.4 Federation Entities Onboarding Process . . . 241
8.4.1 Hierarchical Federation Authority Model . . . 241
8.4.2 Federation Onboarding Prerequisites . . . 242
8.4.3 Key Management Security Requirements . . . 243
8.4.4 Federation Onboarding Procedure . . . 243
8.4.5 IT-Wallet Federation Trust Marks . . . 250

9 X.509 Certificate Management Operations 255
9.1 Federation PKI Architecture . . . 255
9.2 X.509 Certificate Chain Structure and Analysis . . . 256
9.2.1 X.509 Certificate Chain Visualization . . . 256
9.2.2 X.509 Certificate Chain Validation . . . 257
9.3 X.509 Certificate Revocation Management . . . 258
9.3.1 CRL Distribution and Access . . . 258
9.3.2 X.509 Certificate Revocation Verification . . . 258
9.4 X.509 Certificate Management Best Practices . . . 260
9.4.1 X.509 Certificate Validation Integration . . . 260
9.4.2 Diagnostic and Troubleshooting . . . 260
9.4.3 X.509 Certificate Lifecycle Coordination . . . 261
9.5 Key Rotation Procedures . . . 262
9.5.1 Superior Key Rotation . . . 262
9.5.2 Subordinate Key Rotation . . . 263
9.6 Technical Configuration Updates . . . 264
9.7 Technical Federation Exit Procedures . . . 266
9.7.1 Voluntary Exit - Technical Deactivation . . . 266
9.7.2 Supervisory Body Removal - Technical Implementation . . . 267

10 Entities 269

10.1 Wallet Solution . . . 269
10.1.1 Wallet Solution Requirements . . . 269
10.1.2 Wallet Solution Components . . . 276
10.1.3 Wallet Solution Interaction Patterns . . . 279
10.1.4 Wallet Instance . . . 279
10.1.5 Backup and Restore . . . 300
10.1.6 Wallet Provider Entity Configuration . . . 305
10.1.7 Wallet Solution Metadata . . . 308
10.2 Credential Issuer Solution . . . 309
10.2.1 Credential Issuer Requirements . . . 311
10.2.2 Component Details . . . 312
10.2.3 Interaction Patterns . . . 314
10.2.4 Credential Issuer Entity Configuration . . . 314
10.2.5 Credential Issuer Metadata . . . 336
10.3 Relying Party Solution . . . 340
10.3.1 Relying Party Instance . . . 341
10.3.2 Mobile Relying Party Instance . . . 343
10.3.3 Web Relying Party Instance . . . 351
10.3.4 Relying Party Entity Configuration . . . 352
10.3.5 Relying Party Metadata . . . 354
10.4 Authentic Sources . . . 356

11 Digital Credential Management **357**
11.1 Digital Credential Data Model . . . 357
11.1.1 Format-Agnostic Credential Metadata Attributes . . . 357
11.1.2 SD-JWT-VC Credential Format . . . 358
11.1.3 mdoc-CBOR Credential Format . . . 366
11.1.4 Cross-Format Credential Parameters Mapping . . . 379
11.2 PID Data Model . . . 380
11.2.1 PID Data Model in SD-JWT VC Format . . . 382
11.2.2 PID Data Model in mdoc-CBOR Format . . . 387
11.3 IT-Wallet ID Data Model . . . 392
11.3.1 IT-Wallet ID Data Model in SD-JWT VC Format . . . 393
11.4 Digital Credential Lifecycle . . . 398
11.4.1 Credential Transitions . . . 400
11.4.2 Digital Credential Lifecycle in a Batch . . . 402
11.4.3 Credential Lifecycle Management . . . 402
11.4.4 Validity Verification Mechanisms . . . 410

12 Digital Credential Flows **419**
12.1 Digital Credential Issuance . . . 419
12.1.1 Credential Issuance High-Level Flows . . . 419

12.1.2 Credential Issuance Low-Level Flows . . . 422
12.1.3 eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance . . . 457
12.2 Digital Credential Presentation . . . 480
12.2.1 Remote Flow . . . 483
12.2.2 Proximity Flow . . . 511
12.2.3 Identity Matching and Identity Reconciliation . . . 538
12.3 Wallet Metadata Retrieval Flow . . . 541

13 Endpoints **543**
13.1 Wallet Provider Endpoints . . . 543
13.1.1 Federation Endpoint . . . 543
13.1.2 Wallet Solution Nonce Endpoint . . . 544
13.1.3 Wallet Instance Management Endpoint . . . 544
13.1.4 Wallet Instance Attestation Issuance Endpoint . . . 547
13.1.5 Key Attestation Issuance Endpoint . . . 554
13.1.6 e-Service PDND Wallet Provider Catalog . . . 561
13.2 Credential Issuer Endpoints . . . 562
13.2.1 Metadata Endpoints . . . 562
13.2.2 Credential Issuance Endpoints . . . 565
13.2.3 e-Service PDND Credential Issuer Catalog . . . 593
13.3 Relying Party Endpoints . . . 594
13.3.1 Relying Party Remote Flow Endpoints . . . 594
13.3.2 Relying Party Provider Backend Endpoints . . . 598
13.4 Authentic Source Endpoints . . . 602
13.4.1 e-Service PDND Authentic Source Catalog . . . 602
13.5 PDND Signal Hub Endpoints . . . 607
13.5.1 Prerequisites . . . 608
13.5.2 Signal Hub e-Services . . . 609
13.5.3 Signals Processing . . . 613

14 Cryptographic Algorithms **615**

15 Security and Privacy Considerations **619**
15.1 Security Requirements . . . 620
15.1.1 SR-CF-10 and SR-E-10 . . . 620
15.1.2 SR-CF-20 . . . 620
15.1.3 SR-CF-21 . . . 621
15.1.4 SR-E-20 . . . 621
15.1.5 SR-E-30 . . . 621
15.1.6 SR-E-40 . . . 622
15.1.7 SR-I-10 . . . 622

15.1.8 SR-I-20 . . . 622
15.1.9 SR-I-30 . . . 623
15.1.10 SR-I-40 . . . 623
15.1.11 SR-I-50 . . . 623
15.1.12 SR-P-20 . . . 624
15.1.13 SR-P-30 . . . 624
15.1.14 SR-P-40 . . . 625
15.1.15 SR-P-41 . . . 625
15.1.16 SR-P-50 . . . 626
15.1.17 SR-V-10 . . . 626
15.1.18 SR-V-20 . . . 627
15.1.19 SR-W-20 . . . 627
15.1.20 SR-W-30 . . . 628
15.2 Privacy Requirements . . . 628
15.2.1 PR-CF-30 . . . 628
15.2.2 PR-CF-40 . . . 628
15.2.3 PR-E-60 . . . 629
15.2.4 PR-E-70 . . . 629
15.2.5 PR-W-40 . . . 629
15.2.6 PR-W-60 . . . 630
15.2.7 PR-W-70 . . . 630
15.3 Security and Privacy Requirements . . . 630
15.3.1 SPR-E-50 . . . 630
15.3.2 SPR-P-10 . . . 631
15.3.3 SPR-P-60 . . . 631
15.3.4 SPR-P-70 . . . 632
15.3.5 SPR-P-80 . . . 633
15.3.6 SPR-W-50 . . . 633

16 General Log Retention Policies **635**
16.1 ISO/IEC 27001 Logging General Requirements . . . 636
16.1.1 Wallet Provider Log Retention Policy . . . 636
16.1.2 Credential Issuer Log Retention Policy . . . 637
16.1.3 Authentic Source Log Retention Policy . . . 637
16.1.4 Relying Party Log Retention Policy . . . 637
16.1.5 Wallet Instances Logging Features . . . 637

17 Defined Terms and References **639**
17.1 Defined Terms and Acronyms . . . 639
17.1.1 Acronyms . . . 651
17.2 Normative References . . . 652
17.2.1 Official Resources . . . 653

17.3 Technical References . . . 654
17.3.1 Wallet Architecture and Reference Frameworks . . . 655
17.3.2 EUDI Wallet Standards and Technical Specifications . . . 655
17.3.3 Infrastructure of Trust . . . 655
17.3.4 Digital Credential Data Format . . . 655
17.3.5 Digital Credential Issuance . . . 656
17.3.6 Digital Credential Presentation . . . 656
17.3.7 Digital Credential Revocation Check Mechanisms . . . 656
17.3.8 National Data Interoperability Platform Specifications . . . 657
17.3.9 National Digital Identity Platform Specifications . . . 657
17.3.10 Security and Protection Profiles . . . 657

18 How to contribute **659**
18.1 Acknowledgements . . . 659

19 Open Source Releases **663**
19.1 1. General Principles . . . 663
19.2 2. Standard Requirements . . . 663
19.2.1 2.1 Licensing . . . 663
19.2.2 2.2 Code Management & Transparency . . . 664
19.2.3 2.3 Security & Quality . . . 664
19.3 3. Wallet Providers . . . 664
19.3.1 3.1 Mandatory Open Source . . . 664
19.4 4. Credential Issuers and Relying Parties . . . 665
19.4.1 4.1 Integration components . . . 665
19.4.2 4.2 Backend infrastructure . . . 665
19.5 5. Responsible Disclosure and Vulnerability Management . . . 665

20 Appendix **667**
20.1 Mobile Application Instance . . . 667
20.1.1 Mobile Application Instance Initialization . . . 667
20.2 e-Service PDND . . . 675
20.2.1 Requirements and Security Patterns . . . 676
20.2.2 PDND Voucher Issuance . . . 678
20.2.3 Key Retrieval . . . 692
20.2.4 e-Service Usage . . . 697
20.3 PDND e-Service Template . . . 708
20.3.1 PDND Template e-service definition and guidelines . . . 709
20.4 Test Plans . . . 726
20.4.1 Structure of the Test Matrix . . . 727
20.5 Wallet Provider PDND OpenAPI Specification . . . 823

Index 833

This document provides the technical architecture, implementation framework and design requirements to be adopted by IT-Wallet Solutions.

CHAPTER

ONE

INTRODUCTION

Over the last decade, digitalization has radically transformed the way citizens and businesses interact with public and private services, introducing new secure, accessible and user-friendly forms of service access.

In Italy, Decree-Law No. 19 of 2 March 2024, converted, with amendments, by Law No. 56 of 29 April 2024, introduced Article 64-quater of Legislative Decree No. 82 of 7 March 2005, establishing the Italian Digital Wallet System - IT-Wallet System. The IT-Wallet System allows natural or legal persons to access public and private services through the secure presentation of Digital Credential, attesting to entitlements, delegations, characteristics, licenses or qualifications. Article 64-quater also provides for the adoption of one or more implementing decrees (decreti attuativi) to define the rules governing the operation of the IT-Wallet System, including the roles of the entities involved, technical and security requirements, and principles of economic sustainability, of which these Technical Specifications – drafted through an open and collaborative process – form an integral part.

Thanks to the IT-Wallet System, natural and legal persons can directly provide, via their Wallet, the information required for accessing services provided by public and private entities in the form of Digital Credentials. Similarly to a physical wallet, the IT-Wallet can contain identity or document-related data, such as a driver's license or health card, as well as a wide range of verifiable digital information, such as a professional qualification, educational diploma, licence or verifiable attribute.

The main roles in the Wallet ecosystem are listed as follow:

- **Credential Issuers**: parties who issue Digital Credentials to Users;
- **Relying Parties**: parties who request Digital Credentials presentations to the User, for Authentication and authorization purposes;
- **Users**: individuals who own a Wallet Instance and have control over the Digital Credentials they can request, acquire, store, and present to Relying Parties;

In this model, the Credential Issuer (e.g., an educational institution) provides Digital Credentials to the User, who can store them in their Wallet Instance. The Wallet Instance is typically provided as a mobile application on the User's smartphone.

What distinguishes this new approach from previous identity access management systems is that Digital Credentials refer to characteristics, qualities or properties, already authenticated at source. These Digital Credentials can be used by the User without the Credential Issuers being aware of their use. During the use of the Digital Credentials, no usage information is released to third parties as the relationship is exclusive between the User and the Relying Party, in a transparent and informed manner. The development of the IT-Wallet System includes a phased experimentation process, aimed at testing the Wallet and assessing its impact in real-world contexts. This process is designed to validate technical components, user experience elements, and interoperability mechanisms, while ensuring a progressive and controlled adoption of the System. Moreover, it supports the continuous improvement of the IT-Wallet and its gradual alignment with the European Digital Identity Wallet (EUDI Wallet), both in terms of architecture and compliance with evolving European specifications.

Other key elements that characterize this new Digital Identity Wallet paradigm include:

- **Confidentiality and control**: Wallets enable individuals to maintain control over the information provided within the presented Credentials. They can choose what attributes or Credentials to present and to whom;
- **Security**: Wallets leverage cryptographic mechanism for the integrity and the security of exchanged data. This avoids identity theft, fraud, and unauthorized access;
- **Interoperability**: Wallets promote interoperability by enabling different systems and organizations to recognize and verify identities, enabling trusted interactions between individuals, organizations, and even across borders;
- **Efficiency and cost reduction**: Individuals can easily manage their own Credentials, avoid handling multiple identity tokens, and reduce repetitive identity verification processes.

1.1 Scope

These Technical Specifications are intended to complement the Guidelines provided for in Article 64-quater of Legislative Decree No. 82/2005 (CAD). Both the Guidelines and these Technical Specifications, once formally adopted, will become part of the regulatory framework for the IT-Wallet System. They will be periodically updated, where necessary, in light of the results of the experimentation phase, the adoption of new national or European legislative acts, and evolving requirements in terms of security and interoperability. These Technical Specifications pursue two main objectives and represent a core component of the implementation framework for the IT-Wallet System.

The first one is to provide a clear and structured set of recommendations, resources and design requirements related to the IT-Wallet System elements that impact on the User Experience. The document, by distinguishing between mandatory regulatory aspects and good design practices,

aims to provide to public entities and private entities interested in taking part in the IT-Wallet System what is necessary to:

- facilitate the understanding and adoption of the Service Model, increasing the number of potential services and usage opportunities for the User;
- adopt the IT-Wallet System's Visual Identity in order to enhance its reliability and recognizability for the User;
- ensure design consistency across macro-functionalities and single interactions between the User and the service Touchpoints;
- maintain an adequate level of quality, promoting the principles of usability, accessibility and inclusivity.

The second focus is to define the technical architecture and reference framework that will serve as a guideline for all the parties involved in the development of the IT-Wallet System.

Additional documentation, tools and resources - hereinafter defined Official Resources - for the design and development of the IT-Wallet System Technical Solutions will be soon available in the official-resources:Official Resources section and will be periodically updated.

1.2 Normative Language and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3 How to Read the Specification

This specification is designed to fulfil the requirements from multiple stakeholders within the IT-Wallet System. Each role has different responsibilities and scopes, and therefore different information needs. This section provides tailored reading paths to help you navigate efficiently to the content most relevant to the implementation goals.

1.3.1 Specification Structure Overview

The specification is organized into the following major sections:

- **Section** *Introduction* (page 3): Establishes scope, and normative language for the IT-Wallet ecosystem.
- **Section** *Architecture Overview* (page 19): Provides high-level view on the Architecture in terms of governance and operational processes enabled.
- **Section** *Brand Identity* (page 15): Provides the IT-Wallet Brand Identity requirements, guidance on the naming convention and on the application of the visual elements that identify the ecosystem.
- **Section** *User Experience Design* (page 35): Provides design principles and high-level functional requirements to ensure a high-quality User Experience across all stages of interaction between the User and the service.
- **Section** *The Infrastructure of Trust* (page 109): Defines the federation-based trust model, entity relationships, and trust evaluation mechanisms that secure the entire ecosystem.
- **Section** *Entities* (page 269): Comprehensive implementation requirements for each ecosystem participant: Wallet Solutions, Credential Issuers, Relying Parties, and Authentic Sources, including their components, interaction patterns, and configuration requirements.
- **Section** *Digital Credential Management* (page 357): Covers Digital Credential data models and formats, lifecycle management, validity verification, and the Credentials Catalog structure.
- **Section** *Digital Credential Flows* (page 419): Detailed implementation guidance for Digital Credential issuance and presentation workflows, including both remote and proximity interaction flows, and identity matching after presentation.
- **Section** *Endpoints* (page 543): Technical specifications for all API endpoints exposed by each entity type, including federation endpoints and specialized service integrations.
- **Section** *Cryptographic Algorithms* (page 615), *Security and Privacy Considerations* (page 619), and *General Log Retention Policies* (page 635) (**Implementation Support**):

Cryptographic requirements, security and privacy considerations, and log retention policies essential for compliant implementations.

- **Section** *Defined Terms and References* (page 639), official-resources:Official Resources, *How to contribute* (page 659), and *Open Source Releases* (page 663) (**Terminology and References**): Comprehensive terminology, normative references, additional documentation, tools, resources and contribution guidelines.
- **Section** *Appendix* (page 667): Provides supplementary technical details, implementation patterns, and testing frameworks including mobile application instance management, national platform integration specifications, and comprehensive test matrices for ecosystem validation.

1.3.2 Reading Paths by Role

For easier access to the topics covered in these specifications, role-based reading paths are presented below. Each path is organized according to the different stages of onboarding to and participation in the IT-Wallet System, with direct references to the sections most relevant to each role.

The proposed reading paths are intended as guidance and do not replace the need to consult other sections when a broader understanding of the system is required.

For entities interested in addressing multiple roles, it is recommended to deepen all the reading paths related to relevant roles.

Authentic Source

The Authentic Source focuses on securely exposing, managing, and guaranteeing the absolute accuracy and integrity of the authoritative raw data underlying (Q)EAA.

Phase 1: Discovery

To understand the general functioning of the ecosystem, the technical architecture and the Infrastructure of Trust.

- **Section** *Introduction* (page 3): Scope and regulatory context of the IT-Wallet System.
- **Section** *Architecture Overview* (page 19): Overview of the IT-Wallet System architecture in terms of governance and enabled operational processes.
- **Section** *The Infrastructure of Trust* (page 109): Key requirements of the federation-based trust model and trust evaluation mechanisms between entities.
- **Section** *Defined Terms and References* (page 639): Comprehensive terminology, normative references, additional documentation, tools, resources and contribution guidelines.

Phase 2: Design

To understand the requirements and design the features, functionalities and specific characteristics underlying a (Q)EAA to be issued.

- **Section** *User Experience Design* (page 35): Key requirements on how to enable Users to obtain (Q)EAAs, (Q)EAA structures, status and management over time.
- **Section** *Digital Credential Management* (page 357): Technical and functional requirements related to (Q)EAA lifecycle.
- **Section** *PDND e-Service Template* (page 708): Standardized blueprint containing all necessary technical and descriptive metadata for the e-service definition.

Phase 3: Implementation

To implement the technological interfaces required to communicate with Credential Issuers and to manage the entire (Q)EAA data lifecycle.

- **Section** *Authentic Sources* (page 356): Authentic Source role and responsibilities.
- **Section** *e-Service PDND* (page 675): Mandatory integration specifications with the PDND (National Digital Data Platform) and the associated interoperability requirements for publishing an e-service.
- **Section** *Authentic Source Endpoints* (page 602): Key requirements for the implementation of APIs enabling the Credential Issuer to securely and consistently retrieve authoritative data and to manage the data lifecycle through Signal Hub endpoints.
- **Section** *Registry Infrastructure* (page 149): Focus on Registry components of interest to the Authentic Source.
- **Section** *General Log Retention Policies* (page 635): General log retention requirements and specific requirements for Authentic Sources in accordance with ISO/IEC 27001.
- **Section** *Test Plans* (page 726): Guide to set up the test environment and validate backend interactions with the test matrices provided by the ecosystem.

Phase 4: Registration

To become registered as an Authentic Source within the system by completing the administrative and technical procedures required.

- **Section** *Onboarding System* (page 23): Overview of the onboarding system architecture and the Authentic Source registration process.
- **Section** *Entity Onboarding* (page 225): Focus on technical implementation procedures for Authentic Source registration.
- **Section** *X.509 Certificate Management Operations* (page 255): Operational procedures for managing X.509 Certificates within the IT-Wallet federation.

Wallet Provider

The Wallet Provider focuses on designing and developing the Wallet Solution that enables the User to store, manage, and present their PID and (Q)EAAs.

Phase 1: Discovery

To understand the general functioning of the ecosystem, the technical architecture and the Infrastructure of Trust.

- **Section** *Introduction* (page 3): Scope and regulatory context of the IT-Wallet System.
- **Section** *Architecture Overview* (page 19): Overview of the IT-Wallet System architecture in terms of governance and enabled operational processes.
- **Section** *The Infrastructure of Trust* (page 109): Key requirements of the federation-based trust model and trust evaluation mechanisms between entities.
- **Section** *Defined Terms and References* (page 639): Comprehensive terminology, normative references, additional documentation, tools, resources and contribution guidelines.

Phase 2: Design

To understand the User Experience requirements and design the Wallet Solution following common patterns to ensure the usability and accessibility of the solutions.

- **Section** *Brand Identity* (page 15): Overview of the IT-Wallet Brand Identity and indications on assets to be adopted by the Wallet Provider.
- **Section** *User Experience Design* (page 35): Key requirements on Interaction Models and Interface layouts and graphic assets to ensure an effective and seamless User Experience and coherence among Wallet Solutions.
- **Section** *Digital Credential Management* (page 357): Technical and functional requirements related to (Q)EAA lifecycle.

Phase 3: Implementation

To implement the Wallet Solution in line with specific technological standards to ensure the communication between the Wallet Solution and other actors.

- **Section** *Wallet Solution* (page 269): Technical and functional requirements on components, functionalities and lifecycle to configure the Wallet Solution.
- **Section** *Digital Credential Flows* (page 419): Technical and functional requirements on (Q)EAA issuance and presentation flows.
- **Section** *Wallet Provider Endpoints* (page 543): Key requirements for the implementation of the Wallet Provider interfaces (APIs) required for interoperability among entities.

- **Section** *Registry Infrastructure* (page 149): Overview of Registry infrastructure and federation Registry.
- **Section** *Cryptographic Algorithms* (page 615): Selection and implementation of cryptographic standards required to secure keys and transactions.
- **Section** *Security and Privacy Considerations* (page 619): Security and compliance requirements for Wallet Solutions.
- **Section** *General Log Retention Policies* (page 635): General log retention requirements and requirements specific for Wallet Providers, in accordance with ISO/IEC 27001.
- **Section** *Mobile Application Instance* (page 667): Requirements for mobile application instance, related to initialization request and response.
- **Section** *Test Plans* (page 726): Guide to set up the test environment and validate backend interactions with the test matrices provided by the ecosystem.

Phase 4: Registration

To become registered as a Wallet Provider within the system by completing the administrative and technical procedures so that the Wallet Solution is recognized by the system.

- **Section** *Onboarding System* (page 23): Overview of the onboarding system architecture and the Wallet Provider registration process.
- **Section** *Entity Onboarding* (page 225): Focus on technical implementation procedures for Wallet Provider registration.
- **Section** *X.509 Certificate Management Operations* (page 255): Operational procedures for managing X.509 Certificates within the IT-Wallet federation.

Credential Issuer

The Credential Issuer focuses on transforming authoritative raw data from Authentic Sources into (Q)EAAs, and managing their entire lifecycle from issuance to revocation or expiration.

Phase 1: Discovery

To understand the general functioning of the ecosystem, the technical architecture and the Infrastructure of Trust.

- **Section** *Introduction* (page 3): Scope and regulatory context of the IT-Wallet System.
- **Section** *Architecture Overview* (page 19): Overview of the IT-Wallet System architecture in terms of governance and enabled operational processes.
- **Section** *The Infrastructure of Trust* (page 109): Key requirements of the federation-based trust model and trust evaluation mechanisms between entities.

- **Section** *Defined Terms and References* (page 639): Comprehensive terminology, normative references, additional documentation, tools, resources and contribution guidelines.

Phase 2: Design

To understand the requirements and technically design (Q)EAA by structuring metadata to meet technical and regulatory requirements.

- **Section** *User Experience Design* (page 35): Key requirements on (Q)EAA structure, issuance, status and management over time.
- **Section** *Digital Credential Management* (page 357): Technical and functional requirements related to (Q)EAA lifecycle.

Phase 3: Implementation

To develop endpoints based on specific protocols, and to implement (Q)EAA issuance, renewal, revocation and all the technical lifecycle management functions.

- **Section** *Credential Issuer Solution* (page 309): Technical and functional requirements on components and interaction patterns to issue and manage (Q)EAAs lifecycle.
- **Section** *Digital Credential Flows* (page 419): Technical and functional requirements on (Q)EAA issuance and presentation flows.
- **Section** *Credential Issuer Endpoints* (page 562): Key requirements for the implementation of Credential Issuer metadata and authorization endpoints.
- **Section** *Cryptographic Algorithms* (page 615): Selection and implementation of cryptographic standards required to secure keys and transactions.
- **Section** *e-Service PDND* (page 675): Mandatory integration specifications with the PDND (National Digital Data Platform) and the associated interoperability requirements for accessing authoritative data required for (Q)EAA issuance.
- **Section** *Security and Privacy Considerations* (page 619): Security and compliance requirements for implemented solutions.
- **Section** *General Log Retention Policies* (page 635): General log retention requirements and requirements specific for Credential Issuers in accordance with ISO/IEC 27001.
- **Section** *Test Plans* (page 726): Guide to set up the test environment and validate backend interactions with the test matrices provided by the ecosystem.

Phase 4: Registration

To become registered as a Credential Issuer within the system, by completing the administrative and technical procedure so that (Q)EAA issued to the Wallet are officially trusted.

- **Section** *Onboarding System* (page 23): Overview of the onboarding system architecture and the Credential Issuer registration process.

- **Section** *Entity Onboarding* (page 225): Focus on technical implementation procedures for Credential Issuer registration.
- **Section** *X.509 Certificate Management Operations* (page 255): Operational procedures for managing X.509 Certificates within the IT-Wallet federation.

Relying Party

The Relying Party focuses on securely requesting, receiving, and verifying the authenticity and validity of the PID and (Q)EAAs presented by the User to grant access to online and offline services.

Phase 1: Discovery

To understand the general functioning of the ecosystem, the technical architecture and the Infrastructure of Trust.

- **Section** *Introduction* (page 3): Scope and regulatory context of the IT-Wallet System.
- **Section** *Architecture Overview* (page 19): Overview of the IT-Wallet System architecture in terms of governance and enabled operational processes.
- **Section** *The Infrastructure of Trust* (page 109): Key requirements of the federation-based trust model and trust evaluation mechanisms between entities.
- **Section** *Defined Terms and References* (page 639): Comprehensive terminology, normative references, additional documentation, tools, resources and contribution guidelines.

Phase 2: Design

To understand the User Experience requirements and design the verification functionalities necessary to provide the service to the end User.

- **Section** *Brand Identity* (page 15): Overview of the IT-Wallet Brand Identity and indications on assets to be adopted by the Relying Party.
- **Section** *User Experience Design* (page 35): Key requirements on Interaction Models, Interface layouts and graphic assets to ensure an effective and seamless User Experience, and coherence among presentation and verification systems.
- **Section** *Digital Credential Management* (page 357): Technical and functional requirements related to (Q)EAA lifecycle.

Phase 3: Implementation

To implement verification functionalities following specific protocols, to send verification requests to the Wallet and receive a response with the User authorization.

- **Section** *Relying Party Solution* (page 340): Technical and functional requirements on components and functionalities for PID and (Q)EAA verification.

- **Section** *Digital Credential Flows* (page 419): Technical and functional requirements on (Q)EAA presentation and verification flows.
- **Section** *Identity Matching and Identity Reconciliation* (page 538): Rules for identity matching and identity reconciliation after presentation of person identification attributes.
- **Section** *Relying Party Endpoints* (page 594): Key requirements for the implementation of (Q)EAA verification endpoints.
- **Section** *Security and Privacy Considerations* (page 619): Security and compliance requirements for implemented solutions.
- **Section** *General Log Retention Policies* (page 635): General log retention requirements and requirements specific for Relying Parties, in accordance with ISO/IEC 27001.
- **Section** *Test Plans* (page 726): Guide to set up the test environment and validate backend interactions with the test matrices provided by the ecosystem.

Phase 4: Registration

To become registered as a Relying Party within the system, by completing the administrative and technical procedure and becoming a reliable actor when requesting User data.

- **Section** *Onboarding System* (page 23): Overview of the onboarding system architecture and the Relying Party registration process.
- **Section** *Entity Onboarding* (page 225): Focus on implementation procedures for Relying Party registration.
- **Section** *X.509 Certificate Management Operations* (page 255): Operational procedures for managing X.509 Certificates within the IT-Wallet federation.

CHAPTER

TWO

BRAND IDENTITY

The IT-Wallet Brand Identity defines the system's personality and it's expressed through a set of standardized elements that distinguish it, such as its naming and the elements of the Visual Identity. It strengthens and promotes a Trust Model at the basis of the entire ecosystem, by introducing recognizable visual elements.

It is important to strengthen the User awareness and trust in the ecosystem by integrating coherent graphical elements into the Technical Solutions and across the various Touchpoints.

Therefore, to ensure that the Brand Identity is applied consistently and coherently across the different Technical Solutions, and that it supports a high-quality User Experience, Primary Actors MUST comply with the requirements for applying the IT-Wallet System Brand Identity within their Technical Solutions. In particular, all Primary Actors:

- MUST exclusively follow the indications specified in the Brand Manual, indicated in the official-resources:Official Resources section.
- MUST exclusively use the Official Resources that will be made available in the related official-resources:Official Resources folder of these Technical Specifications.

2.1 Naming

The term "IT-Wallet System", as defined by the establishing regulation, is the official name that SHOULD be used in institutional, formal, or legal contexts to refer to the system.

The contracted form "IT-Wallet" is the official Brand name that MUST be used in all communications and disclosures, whether written or verbal, physical or digital, to refer to the "IT-Wallet System".

Below are the requirements for the correct spelling and pronunciation. All those who intend to refer to the IT-Wallet System:

- MUST use of the uppercase letters in the term "IT" and in the initial letters of Wallet and System;

- MUST use of the hyphen (“-”) between ”IT” and ”Wallet”, without any spacing;
- MUST correctly pronounce the term “IT”, phonetically separating the two letters “I” and “IT”. In Italian, the correct pronunciation is /i ti/ and not /it/, while in English the correct pronunciations is /ˌaɪ ˈtiː/ e non /ɪt/, /aɪt/ o /ɪti/.

2.2 Visual Identity

The “IT-Wallet” Brand has its own Visual Identity. All actors within the ecosystem MUST apply it to provide visibility of their participation in the IT-Wallet System but MUST NOT use “IT-Wallet” Visual Identity as a replacement of the Visual Identity of their own organization or their own Technical Solutions.

To ensure proper coexistence between the “IT-Wallet” Visual Identity and the Visual Identity of their organization or Technical Solutions, Primary Actors MUST exclusively use the Official Resources and comply with the indications specified in the Brand Manual.

2.2.1 Logo

The Logo is the official graphical element that ensures immediate recognition of the IT-Wallet System and promotes its reliability. It consists of a symbol and the “IT-Wallet” logotype.



Fig. 2.1: “IT-Wallet” Logo on a light background



Fig. 2.2: “IT-Wallet” Logo symbol on a light background

Below are illustrated the requirements for the use of the Logo. All actors intending to refer to the System:

- MAY use the Logo to refer to the IT-Wallet System but MUST NOT use the Logo to identify a specific Technical Solution;
- MUST the Logo exactly as provided in the Official Resources and MUST comply with the related usage specifications provided;
- MUST use the Logo in the `application/svg+xml` data format;
- MUST NOT alter, distort, modify the Logo, or replace it with unofficial logos;
- MUST maintain the minimum clear space as defined in the Brand Manual to ensure visibility and recognizability. They MUST NOT allow other graphic or textual elements interfere with this space;
- MUST NOT resized the Logo below the minimum dimensions specified in the Brand Manual, to maintain legibility across formats and devices;
- MUST NOT place the Logo on backgrounds that compromise its visibility or legibility. They MUST ensure an adequate contrast between the Logo and the background, in accordance with the Brand Manual;
- MAY displayed the Logo alongside other actors' logos, trademarks, or symbols, in accordance with the Brand Manual;
- MAY use the Logo symbol as a simplified version of the Logo, in accordance with the Brand Manual;
- MAY use the Logo symbol within a shape or a button to reference the connection to IT-Wallet, such as in Engagement Buttons (see *Authentication Button* (page 90)), in accordance with the Brand Manual.

For further information on the Logo and its related symbol refer to the Brand Manual, indicated in the official-resources:Official Resources section. The Logo Official Resources are available in the related official-resources:Official Resources section of these Technical Specifications.

2.2.2 Trust Mark

The Trust Mark is the official graphic element that guarantees the belonging of Primary Actors and their respective Technical Solutions to the IT-Wallet System.

Below are illustrated the requirements for the use of the Trust Mark, that apply to both physical and digital contexts (e.g., websites, apps, documents, printed materials, videos, etc.). All Primary Actors:

- MUST use the Trust Mark exclusively to indicate their participation in the IT-Wallet System and MUST NOT use it for any other purpose;



Fig. 2.3: Trust Mark on a light background

- MUST display the Trust Mark only in the Technical Solutions that have successfully completed the registration process;
- MUST display the Trust Mark in a clear and visible manner to the User from the first moment they access the Technical Solution and MUST make it available for consultation within the Technical Solution;
- MUST exclusively use the Trust Mark provided in the Official Resources and MUST comply with the related usage specifications provided in the official-resources:Brand Manual;
- MUST NOT alter, distort, modify the Trust Mark, or replace it with unofficial elements;
- MUST NOT resize the Trust Mark below the minimum dimensions specified in the official-resources:Brand Manual to maintain legibility across formats and devices;
- MUST maintain the minimum clear space as defined in the official-resources:Brand Manual to ensure visibility and recognizability and MUST NOT allow other graphic or textual elements to interfere with this space;
- MUST NOT place the Trust Mark on backgrounds that compromise its visibility or legibility. They MUST ensure adequate contrast between the Trust Mark and the background, in accordance with the official-resources:Brand Manual;
- MAY display the Trust Mark alongside other actors' logos, trademarks, or symbols, in accordance with the Brand Manual.

For further information on the Trust Mark, refer to the Brand Manual, indicated in the official-resources:Official Resources section. The Trust Mark Official Resource is available in the official-resources:Official Resources section of these Technical Specifications.

CHAPTER

THREE

ARCHITECTURE OVERVIEW

The IT-Wallet System is a federated ecosystem that enables secure Digital Identity management and Digital Credential exchange for citizens and organizations. The IT-Wallet ecosystem is built on a multi-layered architecture, where governance bodies establish and maintain the trust infrastructure, Primary Actors implement and operate the technical solutions, and external systems provide additional services.

The following diagrams depict the IT-Wallet architecture overview.

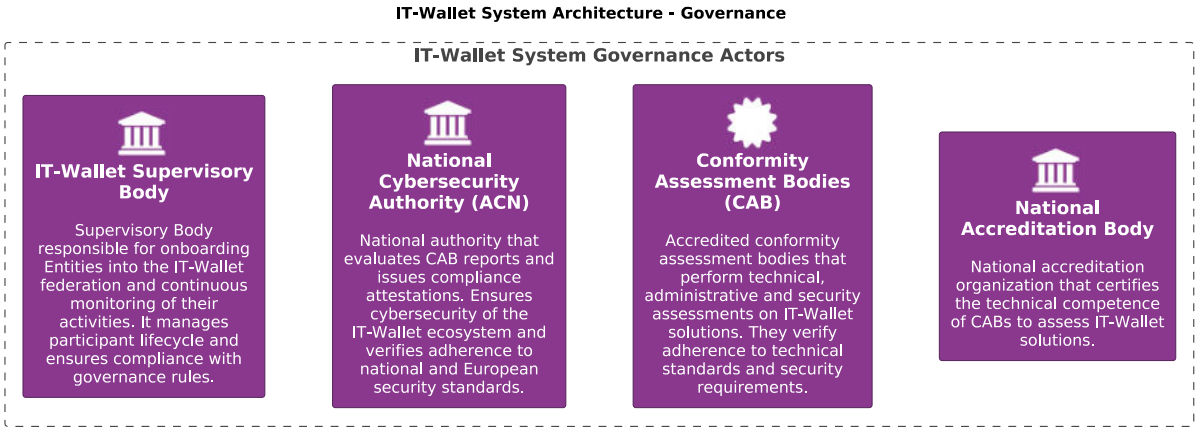


Fig. 3.1: IT-Wallet architecture overview - Governance.

The governance level requires all participants in the ecosystem to comply with security and technical standards and requirements.

The primary operational layer implements the core Digital Credential lifecycle through coordinated interactions among specialized technical solutions, so that all Credential operations maintain security and privacy standards while enabling seamless user experiences.

External systems provide services that connect the IT-Wallet ecosystem to the national digital infrastructure, enabling interoperability with existing government services and data sources.

These three architectural layers work together to enable secure Digital Credential operations. The federated nature of this architecture require systematic onboarding processes to establish trust relationships between participants and standardise Credential semantics through the

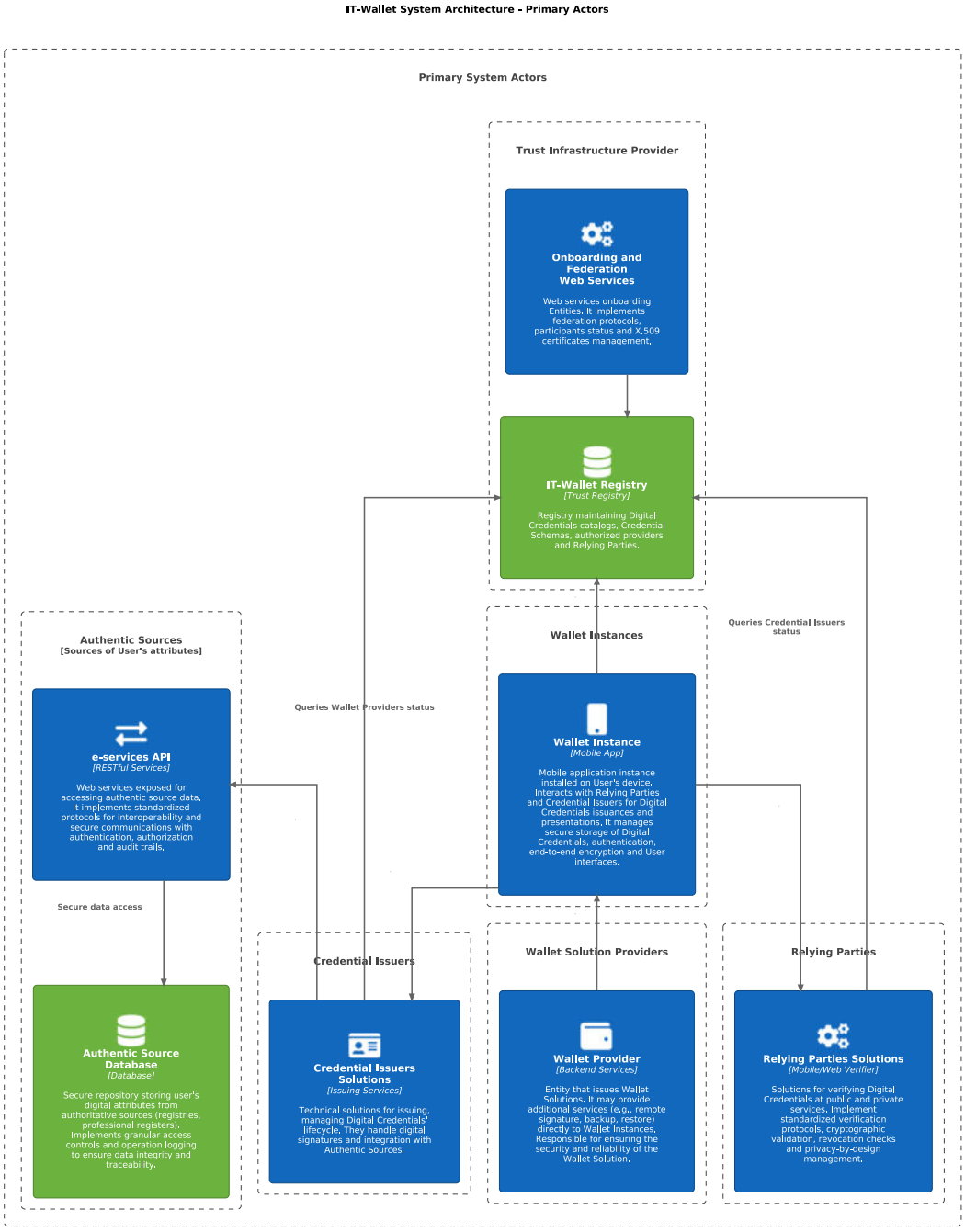


Fig. 3.2: IT-Wallet architecture overview - Primary Actors.

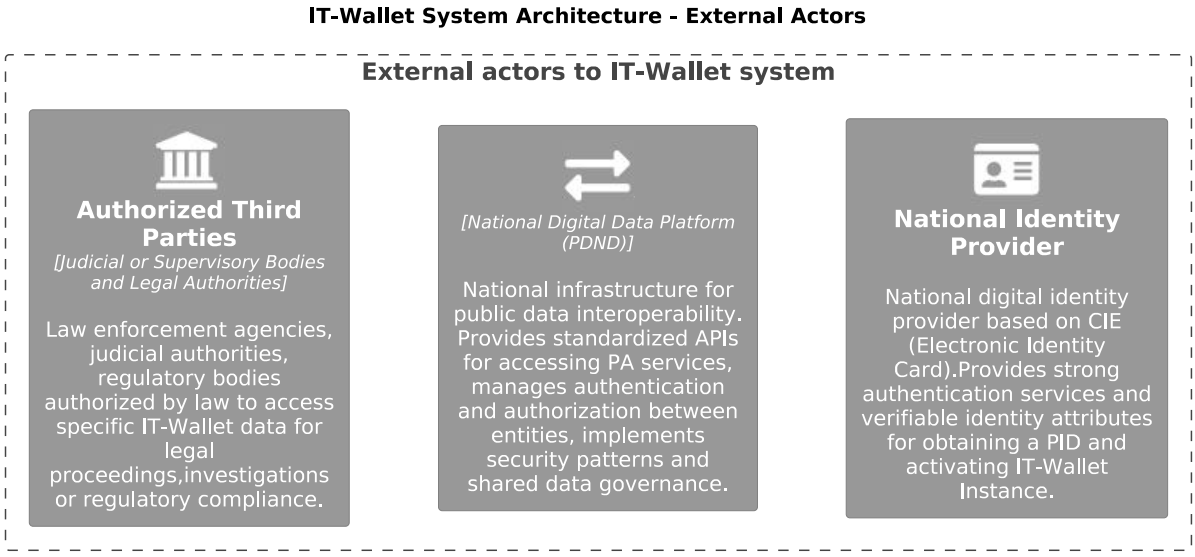


Fig. 3.3: IT-Wallet architecture overview - external Actors.

centralised Claims Registry and Taxonomy components, as detailed in *Registry Infrastructure* (page 149).

The architecture enables the following core interaction processes:

1. **Entity Onboarding and Federation**: Only qualified entities can participate in the federation through systematic onboarding that ensures compliance and establishes cryptographic trust relationships. The onboarding system includes a **Claims Registry** and **Taxonomy** that standardize semantic definitions for all Credentials, enabling interoperability and governance. Entity authorization follows policy-based approach that grants specific operational scopes based on organizational characteristics and regulatory compliance. See Section *Onboarding System* (page 23) and *Registry Infrastructure* (page 149).

2. **Credential Discovery and Catalog Management**: The ecosystem supports dual Credential discovery mechanisms - public discovery through the *Digital Credentials Catalog* (page 179) for general-purpose Credentials, and private discovery via direct Credential Offers for specific use cases. The catalog is automatically populated based on Supervisory Body policies from registered Claims Registry and Taxonomy definitions.

3. **Credential Issuance**: Credential Issuers coordinate with Authentic Sources via standardised APIs to request verified User attributes, creating Digital Credentials based on authoritative data. Issuance can proceed for both Credentials discovered within the Credential Catalog and Credential provided through Credential offer flows.

4. **Credential Storage and Management**: IT-Wallet Solutions receive and manage Digital Credentials on User devices, enabling secure Credential lifecycle management.

5. **Credential Presentation and Verification**: Users present Digital Credentials to Relying Parties, which verify claims through cryptographic validation and status verification.

Relying Parties operate within authorized scopes that define which Credential types and purposes they can request, with validation against the *Claims Registry* (page 152) for schema verification.

CHAPTER

FOUR

ONBOARDING SYSTEM

The onboarding system registers operational entities (PID Providers, Attestation Providers, Relying Parties, Wallet Providers, and Authentic Sources) so that they can participate in the IT-Wallet ecosystem with clearly defined authorizations and trust relationships.

4.1 Onboarding System Architecture

The onboarding service MUST provide specialized onboarding processes that match the operational characteristics and regulatory obligations of different participant types.

All Primary Actors MUST undergo administrative registration for legal and regulatory compliance, followed by specialized technical registration processes that MUST reflect their operational roles in the Digital Credential ecosystem.

1. **Administrative Registration**: All entities (Authentic Sources, Relying Parties, Wallet Providers, Credential Issuers) MUST complete initial administrative registration that validates their legal standing, regulatory compliance, and organizational eligibility to participate in the IT-Wallet ecosystem.
2. **Technical Registration**: Following administrative approval, entities complete technical registration through specialized pathways:
    a. **Authentic Sources**: Declare their available claims from the Claims Registry and specify intended verification purposes and organization type (public or private).
    b. **Credential Issuers**: Select Authentic Sources based on required claims, request integration approval (except for regulatory mandates), and register Credential types with automatic catalog publication per Supervisory Body policy.
    c. **Other Federation Entities** (Relying Parties, Wallet Providers): Undergo federation-based registration for cryptographic trust establishment.
3. **IT-Wallet Registry Integration**:

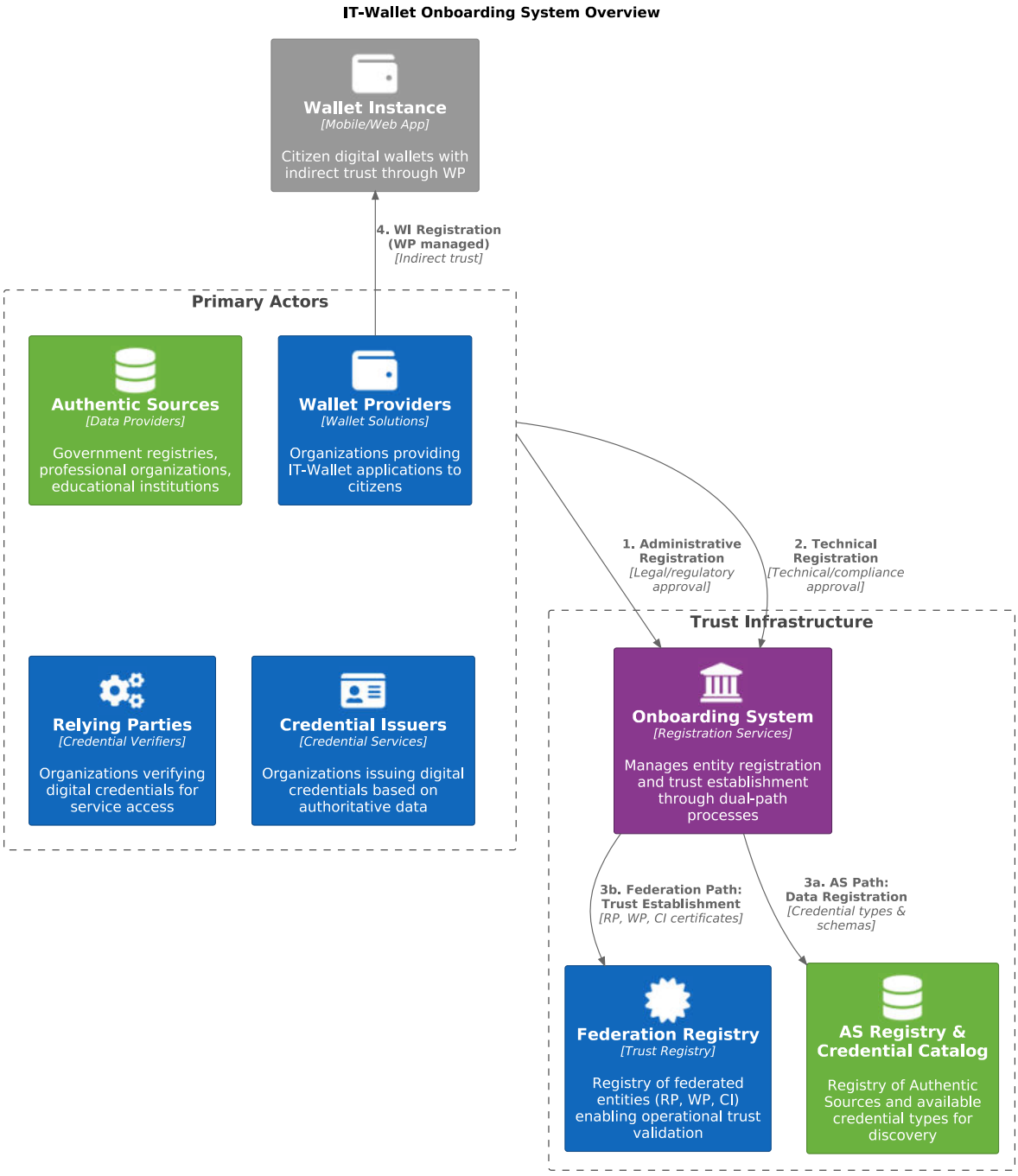


Fig. 4.1: IT-Wallet Onboarding System Overview.

a. **Claims Registry and Taxonomy Integration**: Claims Registry provides standardized data definitions for individual Credential attributes, while Taxonomy defines hierarchical classification (domains, classes, purposes) that are then referenced in the Digital Credentials Catalog for specific Credential implementations. All participants leverage these registries for capability declarations and issuance/verification requirements.

b. **AS Registry Integration**: Authentic Sources registered with their declared claims and capability, enabling CI discovery and coordination.

c. **Federation Registry Integration**: Operational entities included for trust validation during Credential operations.

d. **Catalog Integration**: Credential types automatically published in *Digital Credentials Catalog* (page 179) based on Supervisory Body policies for public discovery eligibility.

4. **Wallet Instance Registration**: Wallet Instances are registered indirectly through Wallet Providers, establishing user-level Credential management capabilities. Technical details are provided in *Wallet Instance Initialization and Registration* (page 285).

4.2 Authentic Source Registration Process

Authentic Source registration allows data providers to establish their authoritative role in the Digital Credential ecosystem through the registration of their data capabilities and standardized access mechanisms based on the Claims Registry and Taxonomy classifications.

Authentic Source entities MUST undergo registration procedures that validate their data authority, declare their available claims from the standardized Claims Registry, and establish technical integration mechanisms. Authentic Source entities specify intended use cases (formally `purposes`) that determine catalog eligibility per Supervisory Body policies.

Public and Private Authentic Sources MUST leverage PDND integration to provide government data through standardized national infrastructure.

This pathway MUST assure data quality standards and establish audit trails for all data provisioning activities.

AS-CI Coordination Process: Following AS registration, Credential Issuers identify suitable AS entities through the AS Registry and request integration authorization during the administrative registration phase. For regulatory mandates, authorization MUST be automatic. Otherwise, Authentic Sources entities evaluate and authorize Credential Issuers requests based on business and technical criteria. Following administrative authorization, technical integration procedures establish the operational data access relationships before Credential type catalog publication.

Successfully registered Authentic Sources MUST be included in the AS Registry with their declared claims and capability. Credential types MUST become publicly discoverable in the *Digital Credentials Catalog* (page 179) only after successful AS-CI integration and Supervisory Body policy approval for catalog eligibility.

Technical implementation procedures for Authentic Source registration are provided in *Authentic Sources Registration Process* (page 227).

4.3 Federation Onboarding Process

Federation onboarding, including administrative and technical registration processes, establishes the trust relationships and compliance checks that enable all entities (PID Providers, Attestation Providers, Relying Parties, Wallet Providers, and Authentic Sources) to participate in the Credential lifecycle. Operational entities (PID Providers, Attestation Providers, Relying Parties, and Wallet Providers) MUST complete onboarding that includes administrative eligibility verification and technical validation. For Wallet Providers, this includes conformity assessment of the Wallet Solution and subsequent Trusted List notification flows.

For PID Providers, Attestation Providers, and Relying Parties, the federation onboarding process typically includes:

- **Entity Registration**: Collection of core data (identification, entitlements, service supply points, and cryptographic public keys and certificates) needed to authorize entities and describe their capabilities.
- **Certificate Issuance**: Issuance of certificates that authenticate entities, reference the registry for entitlement verification, support certificate transparency, and describe registration status for Relying Parties and Credential Issuers.
- **Registry Publication**: Publication of all registered entities in the registry, with an online API (see *Trust Infrastructure and Registry Integration* (page 111)) that can be used to verify entity registration, and requested attributes.

4.3.1 Onboarding of Relying Party Intermediaries

Relying Party Intermediaries use a dedicated onboarding process that grants the status of recognized Intermediary and the authority to onboard subordinate Relying Parties. In the implementation profile described by these technical specifications, a Relying Party Intermediary matches an OpenID Federation Intermediate Entity (see the definition of **IT-Wallet Intermediary**). The process includes:

- **Administrative registration**: The Intermediary submits organizational and legal documentation demonstrating its eligibility to act on behalf of Relying Parties pursuant to Art. 5b(8) of the eIDAS2 Regulation (EU_2024_1183).

- **Technical registration**: The Intermediary registers its federation keys and endpoints, enabling it to publish its own Entity Configuration and issue Subordinate Statements for subordinate Leaves.
- **Issuance of the Intermediary Trust Mark**: Upon completion of the onboarding process, the Trust Anchor issues to the Intermediary a Trust Mark with identifier `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier_intermediary`. This Trust Mark is the verifiable evidence of the Intermediary's recognized role in the federation.
- **Authorization to issue Trust Marks**: The Trust Anchor updates its own Entity Configuration by including the Intermediary in the `trust_mark_issuers` attribute for the `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier` identifier, authorizing the Intermediary to issue Trust Marks to its affiliated Relying Parties.

The affiliated RP's Entity Configuration MUST include the Trust Mark issued by the Intermediary and MUST set `authority_hints` using the Intermediary's URL identifier.

4.4 Entity Lifecycle Management

Following successful onboarding, entities require ongoing lifecycle management to maintain operational status and compliance within the IT-Wallet ecosystem. Lifecycle management encompasses administrative updates, technical modifications, and federation exit processes that accommodate changing organizational and operational requirements.

4.4.1 Ongoing Operations Management

Entities MUST maintain current administrative and technical information to ensure federation participation and compliance.

Administrative Updates

Organizations MAY update administrative information through standard registry channels, such as:

- **Legal Entity Changes**: Company name changes, organizational restructuring, legal status modifications.
- **Contact Information**: Updates to official contact channels and responsible personnel.
- **Regulatory Status**: Changes in licenses, certifications, or regulatory compliance status.
- **Service Scope**: Modifications to service offerings or user base characteristics.

Administrative updates MUST follow standard governance processes and SHOULD NOT affect technical federation operations.

Technical Configuration Management

Technical updates affecting federation protocol operations require coordinated procedures, including:

- **Certificate Management**: Regular certificate renewal, emergency replacement, revocation handling.
- **Infrastructure Changes**: Endpoint updates, service migrations, capacity modifications.
- **Compliance Updates**: Security standard updates, policy changes, audit requirements.
- **Capability Modifications**: Adding or removing supported protocols, Credential types, or service features.

Technical updates MUST be validated by the Supervisory Body to maintain federation trust relationships and operational integrity.

4.4.2 Federation Exit and Removal Processes

Entities MAY exit the federation voluntarily or be removed by the Supervisory Body for compliance or security reasons.

Voluntary Federation Exit - Organizations MAY choose to exit the federation for business or operational reasons, including:

- **Business Changes**: Organizational restructuring or service discontinuation.
- **Technical Migration**: Moving to alternative technical solutions or providers.
- **Regulatory Changes**: Changes in regulatory environment or compliance requirements.

Voluntary exit MUST require coordination with dependent entities and proper handling of existing Credentials and user relationships.

Supervisory Body Removal - Supervisory Body MAY initiate entity removal for:

- **Compliance Violations**: Failure to maintain regulatory compliance or federation policy adherence.
- **Security Incidents**: Compromise of security infrastructure or failure to maintain security standards.
- **Operational Failures**: Persistent technical failures that affect federation security.
- **Policy Violations**: Violations of federation operational policies or agreements.

Removal processes MAY include investigations, remediations, and appeal procedures where appropriate.

Warning

For critical security incidents or immediate threats to federation integrity, the Supervisory Body MAY implement emergency suspension with immediate effect.

4.4.3 Lifecycle Coordination Requirements

Lifecycle management in the IT-Wallet ecosystem needs coordination between multiple participants to keep operations working and maintain trust relationships. The coordination framework covers three main areas:

- **Stakeholder communication:** When entities need to make changes that impact IT-Wallet ecosystem operations, they MUST inform the Supervisory Body and relevant federation participants in advance.
- **Registry synchronization:** When one entity makes changes that affect other entities, all registry systems MUST be properly updated ensuring that all changes are logged securely with timestamps and reasons.
- **Business continuity assurance:** Entities SHOULD balance between making necessary updates and keeping their obligations to users and regulations. This includes ensuring that the service is as available as possible, handling personal data correctly during changes, and staying compliant with legal requirements even in emergency situations.

Technical procedures and specific compliance requirements for lifecycle management are detailed in the Section *Entity Onboarding* (page 225).

4.5 Onboarding Journey Maps

The following journey maps provide a detailed view of the onboarding experience from the perspective of each entity type and their operators. These visual representations help organizations understand the specific steps, requirements, and interactions they will encounter during their onboarding process.

Each journey map shows the complete process from initial planning to final registry integration, highlighting critical touchpoints with the Supervisory Body and dependencies between different entity onboarding processes.

4.5.1 Ecosystem Overview

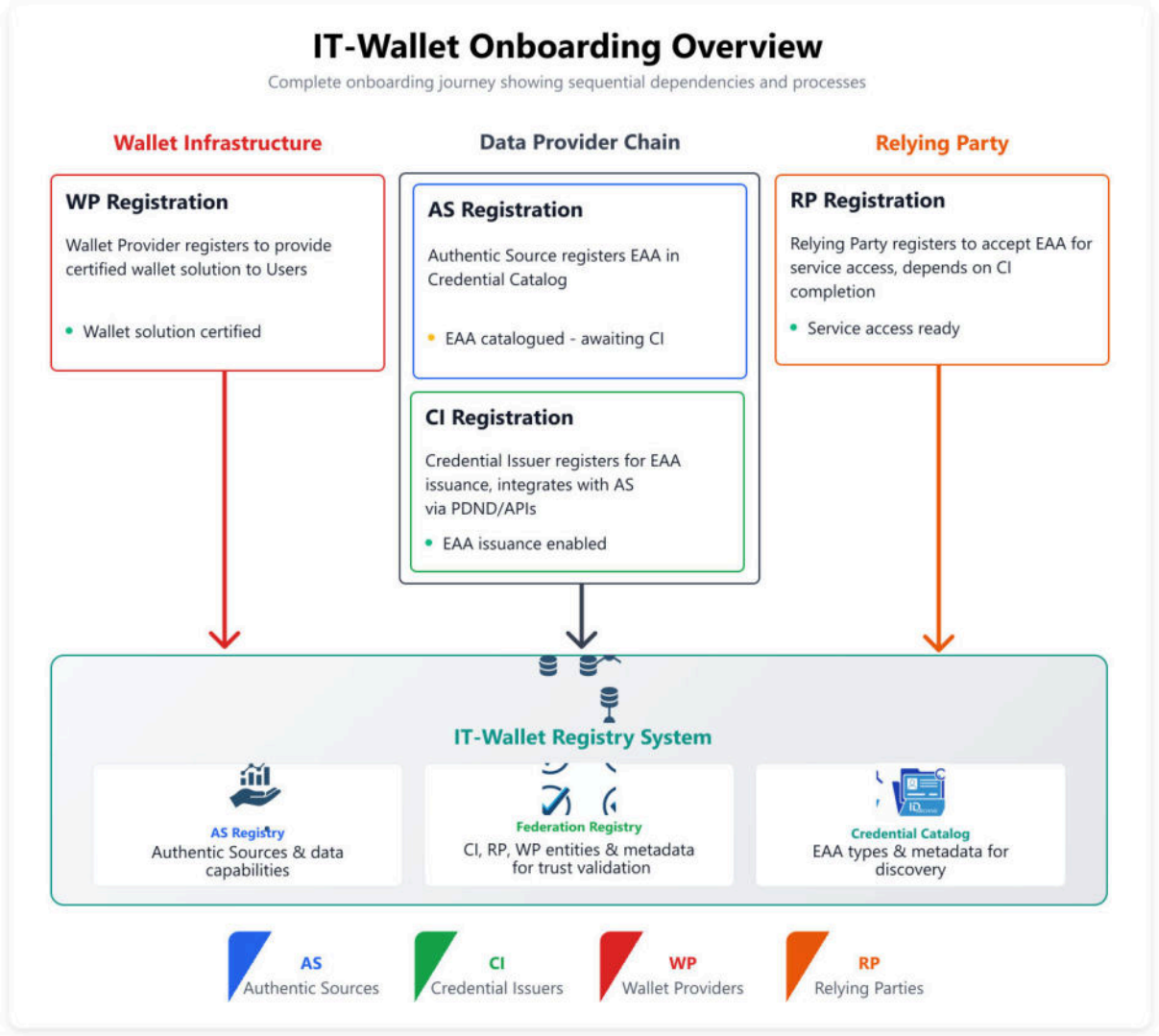


Fig. 4.2: Complete ecosystem onboarding showing the main processes

The IT-Wallet Registry system coordinates all registrations through five main components:

1. **Claims Registry** for standardized semantic definitions of individual Credential attributes.
2. **AS Registry** for data sources and capabilities.
3. **Federation Registry** for operational trust relationships.
4. **Digital Credentials Catalog** (see *Digital Credentials Catalog* (page 179)) for Credential type discovery.
5. **Taxonomy** for hierarchical classification organizing Credentials by domain and purpose.

The following journey maps illustrate two distinct Credential scenarios:

- **Public Catalog Scenario**: Mobile Driving License (mDL) provided for a public discovery via Credential Catalog.
- **Private Credential Scenario**: Corporate Employee Badge from Company (Private AS, provided for a public discovery via Credential Catalog but obtainable via Credential Offer only).

4.5.2 Authentic Source Operator Journey

From the Authentic Source operator perspective, the onboarding process begins with evaluating existing data capabilities against the standardized Claims Registry and Taxonomy classifications, determining which user attributes can be made available as a Digital Credential. The operator submits a registration request to the Supervisory Body, declaring specific claims from the Claims Registry with the Taxonomy domains and intended purposes.

Example - Public Authentic Source (mDL Scenario):

- **Claims Declaration**: Selects standardized claims (`given_name`, `family_name`, `driving_privileges`, etc.) from Claims Registry.
- **Taxonomy Classification**: Domain `MOBILITY_TRAVEL`, Purpose `DRIVING_RIGHTS`.
- **Use Case**: Public service - driving authorization verification.
- **Integration**: e-Service PDND integration following government standards (see *e-Service PDND* (page 675)).
- **Catalog Outcome**: mDL becomes publicly discoverable after CI integration.

Example - Corporate AS (Employee Badge Scenario):

- **Claims Declaration**: Selects claims (`given_name`, `family_name`, `employee.job_title`, etc.) from Claims Registry.
- **Taxonomy Classification**: Domain `MEMBERSHIP`, Purpose `ASSOCIATION`.
- **Use Case**: Private corporate access control.
- **Integration**: e-Service PDND integration following government standards (see *e-Service PDND* (page 675)).
- **Catalog Outcome**: Badge becomes publicly discoverable but it is available for the issuance only via Credential Offer.

Critical phases include administrative verification by the Supervisory Body (which involves regulatory compliance checks outside the technical scope) and technical validation. The process concludes with registration in the AS Registry, making the declared claims discoverable by Credential Issuers for integration requests. The integration requests can be also send by the Authentic Sources to a specific Credential Issuers.

Warning

Important dependency: Declared claims in AS Registry remain unavailable to end users until a Credential Issuer completes registration, integration approval, and technical implementation. Catalog publication depends on Supervisory Body policies for public discovery eligibility.

4.5.3 Credential Issuer Operator Journey

Credential Issuer operators start by discovering available Authentic Source entities through the AS Registry and evaluating integration feasibility based on required claims. The registration request specifies which Credential types they intend to issue, select appropriate Authentic Source entities, and demonstrate technical capability to access the required data sources. Alternatively, the Credential Issuer operators receive directly the integration request by the Authentic Source.

Example - mDL (Public Scenario):

- **AS Discovery** or **AS Integration Request**: Identifies the Authentic Source providing mDL attributes in AS Registry with required driving license claims. Alternatively, Credential Issuer receives the integration request directly by mDL Authentic Source.
- **Integration Request**: Automatic approval due to regulatory mandate.
- **Technical Setup**: e-Service PDND integration following government standards (see *e-Service PDND* (page 675)).
- **Catalog Publication**: mDL automatically published in the Credential Catalog.
- **User Access**: Citizens discover mDL through a public catalog in Wallet.

Example - CI for Employee Badge (Private Scenario):

- **AS Discovery** or **AS Integration Request**: Identifies the Authentic Source in AS Registry with employee access claims. Alternatively, Credential Issuer receives the integration request by the Authentic Source.
- **Integration Request**: Requires recipient approval.
- **Technical Setup**: e-Service PDND integration following government standards (see *e-Service PDND* (page 675)).
- **Catalog Publication**: Badge automatically published in the Credential Catalog.
- **User Access**: Employees receive badges only via direct Credential Offer from company systems.

Following successful integration testing and Authentic Source approval, the Credential Issuer is registered in the Federation Registry as a trusted Entity, enabling actual Credential issuance to end-users.

Warning

This step activates Credential availability for end-users. Public and Private Credentials become discoverable through the catalog, but the Private Credentials will be available only via direct Credential Offers.

4.5.4 Wallet Provider Operator Journey

Wallet Provider operators follow an independent onboarding path that focuses on application certification and security validation. The process highlights the development and certification of wallet applications that can securely store and manage Digital Credentials for citizens.

A key technical requirement involves implementing Wallet integrity and authenticity check mechanisms. These checks enable the Wallet to obtain a Wallet Instance Attestation, which serves as proof of the Wallet's security and compliance status during Credential operations.

The certification process includes security evaluation, covering wallet architecture, data protection mechanisms, and user privacy features. Successfully certified wallet providers are registered in the Federation Registry and can distribute their applications through app stores.

4.5.5 Relying Party Operator Journey

Relying Party operators begin by identifying which EAA types are required for their specific services and evaluating integration complexity with existing authentication systems. The registration request provides evidence of legitimate needs for accessing specific Credential types and User attributes, outlining the intended service use cases and organizational characteristics that justify Credential access.

Example - Car Rental Service (Private RP):

- **Business Classification**: ATECO code 77.11 (car rental services).
- **Authorization Request**: Driving authorization verification for rental eligibility.
- **Claims Requirements**: `given_name`, `family_name`, `driving_privileges`, etc., from mDL.
- **Use Case Justification**: Legal obligation to verify valid driving license before vehicle rental.
- **Authorization Scope**: Granted access to `MOBILITY_TRAVEL` domain, `DRIVING_RIGHTS` purpose.

Example - Municipal Services (Public RP):

- **Organization Type**: Public Administration with IPA code requirement.
- **Authorization Request**: Citizen identity verification for municipal services access.
- **Claims Requirements**: `given_name`, `family_name`, `personal_administrative_number` from PID.
- **Use Case Justification**: Public service delivery requiring citizen identification.
- **Authorization Scope**: Granted broader access reflecting public service mandate.

Example - Corporate Access Control (Private RP):

- **Business Classification**: ATECO code 62.01 (computer programming activities).
- **Authorization Request**: Employee verification for corporate facility access.
- **Claims Requirements**: `given_name`, `family_name`, `employee.job_title` from Corporate Employee Badge.
- **Use Case Justification**: Workplace security requiring employee identification and access control.
- **Authorization Scope**: Granted access to `MEMBERSHIP` domain, `ASSOCIATION` purpose.
- **Credential Discovery**: Badge available only via private Credential Offer (non-eligible for Credential Catalog).

Technical integration focuses on developing authentication flows that can verify Digital Credentials presented by Users. This includes implementing cryptographic verification mechanisms and establishing secure communication channels with the federation infrastructure.

Service authorization by the Supervisory Body MUST involve policy-based evaluation that considers organizational type (private vs public administration), business sector classification, and legitimate service requirements. The authorization process grants specific operational scopes that define which Credential domains and purposes the Relying Party can request. Following approval, the Relying Party is registered in the Federation Registry with clearly defined authorization boundaries for Digital Credentials and User's attributes acceptance.

4.6 User Experience Journey

When all entity onboarding processes are successfully completed, Users can discover and install certified Wallet Instances, obtain available Digital Credentials and present their Digital Credentials to registered Relying Parties (see *Functionalities Overview* (page 36)).

This seamless integration depends on all relevant entities having completed their respective onboarding journeys.

CHAPTER

FIVE

USER EXPERIENCE DESIGN

5.1 Design Principles

The IT-Wallet System adheres to the Digital Identity Wallet Paradigm, aiming to provide Users with a simple, fast, and more secure experience when accessing services.

The Wallet Solution serves as the primary Touchpoint, enabling Users to manage and use their personal identification data and Attributes when interacting with public or private entities, in both physical and digital contexts. However, the overall service depends on the interactions of a network of Touchpoints and Technical Solutions, that support its delivery and directly impact the User Experience.

To ensure service quality from a User Experience perspective, Primary Actors, both public and private entities, MUST ensure the usability and accessibility of their Technical Solutions while aligning with the distinctive elements of the IT-Wallet System. Specifically:

- **Usability**: Technical Solutions MUST be designed and maintained to meet high usability standards, to facilitate service adoption and reduce the need for assistance. Public entities MUST adhere to [GL_DESIGN], whereas private entities MAY refer to it as a best practice.
- **Accessibility**: Technical Solutions MUST be designed and maintained to meet high accessibility standards to ensure service access regardless of individual abilities, technological skills, or external and contextual constraints. Public entities MUST adhere to [REF_ACCESSIBILITY], whereas private entities MUST comply with applicable regulations.
- **Consistency**: Technical Solutions MUST be designed and maintained in adherence to the IT-Wallet System's Brand Identity and User Experience functional requirements in this document, to promote the recognizability of components, to ensure the overall system consistency, and to minimize the User's cognitive load.

5.2 Functionalities Overview

The IT-Wallet System provides Users with a simpler, faster, and more secure way to access services. This service is delivered through the use of a Wallet Solution, whose User Experience is structured into three main phases: pre-use, use, and post-use.

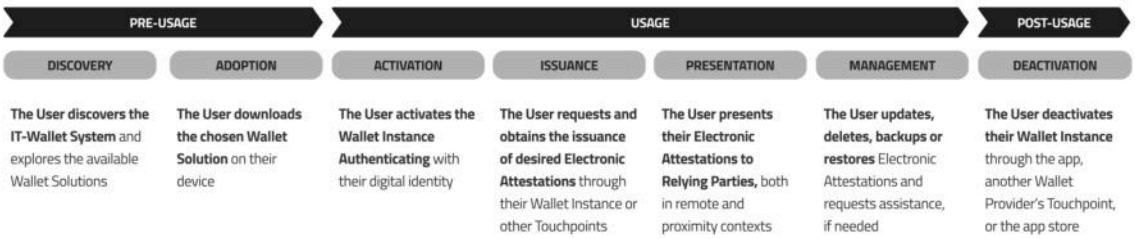


Fig. 5.1: User Experience phases of Wallet usage

The following sections focus on the usage and post-usage phases. They define the functional requirements supporting the User Experience for the activation, acquisition, presentation, management, and deactivation phases, along with interaction requirements related to error management, assistance requests, and feedback collection.

Additional documentation and resources are provided in the official-resources:Official Resources section.

The Official Resources include recommendations on the required User-Wallet Instance interactions and design best practices that promote consistency across different Wallet Solutions in terms of how functionalities are accessed and used.

To ensure a correct and consistent implementation, Primary Actors:

- MUST use the official-resources:Official Resources and MUST comply with all related usage specifications provided;
- MAY choose from the available configurations provided. Primary Actors MUST ensure the correct use of atomic components, such as the *Engagement Button* (page 49) or the *Authentication Button* (page 90) ;
- MUST keep used resources up to date, in line with the latest available version of the official-resources:Official Resources.

5.3 Activation of the Wallet Instance

Activation enables the User to access the Wallet Solution's functionalities for securely obtaining, presenting, and managing their Electronic Attestations. The activation process involves User Authentication with the Wallet Instance using their digital identity (see *Wallet Instance Lifecycle* (page 279)). After Authentication, the User obtains a person identification Attestation — either the EUDI **PID** or the national **IT-Wallet ID** — as described in *Focus on Person Identification Attestations* (page 38).

Below are the User Experience requirements that the Wallet Provider MUST guarantee via their Wallet Solution:

- The User downloads the Wallet Solution onto their device to generate their Wallet Instance;
- The User sets an unlock PIN for their Wallet Instance if one has not been previously set in the app. In addition to the PIN, the User can decide to use their own unlock mechanism used within the device and managed at the operating system level (e.g., biometric authentication) as an alternative to the PIN. The User uses the unlock method whenever an authorization is required to ensure security and protect their information;
- The User reviews all relevant information regarding the activation process and service usage. Additionally, the User reads any policy from the Wallet Provider and from the issuer of the person identification Attestation (PID Provider or IT-Wallet ID provider) and/or the service's terms and conditions. The User gives their consent to proceed or declines to cancel the operation;
- The User selects an Authentication option from those available;
- The User completes the Authentication flow with the National Identity Provider's service;
- The User receives confirmation of the Authentication process outcome. If successful the User views a preview of the person identification Attestation (PID or IT-Wallet ID). The User confirms the previewed information to proceed, or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User receives confirmation of the successful activation of the Wallet Instance.

The Wallet Provider MUST allow the User to remove a person identification Attestation issued within the Authentication phase. In addition, the relevant issuer (PID Provider or IT-Wallet ID provider) SHOULD allow the User to revoke that Attestation through a specific Touchpoint. The Wallet Provider MUST allow the User to always have the option to request the deactivation of their Wallet Instance, even in the absence of the device on which it was installed. For further details, please refer to the *Deactivation of the Wallet Instance* (page 101) and *Management of Electronic Attestations* (page 94) sections.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

5.3.1 Focus on Person Identification Attestations

This section defines User Experience requirements for person identification Attestations held in the Wallet Instance: the EUDI **PID** and the national **IT-Wallet ID** (Electronic Attestation of Person Identification Data with national scope). Both convey a verified minimum set of information about the User identity (see *Digital Credential Data Model* (page 357)), but they differ in legal scope, issuer type, Wallet Instance lifecycle impact, and use cases for presentation supported.

Below are illustrated the User Experience requirements to ensure a uniform and consistent usage and display of the personal identification Attestation (PID and/or IT-Wallet ID). The Wallet Provider:

- MUST correctly display the person identification Attestation across all devices, ensuring a consistent experience on screens of varying sizes;
- MUST display the Attestation status if different from valid, to provide transparency on its lifecycle, and MAY display it if valid. Specific details about an invalid status MAY be provided (e.g., the reason why the Attestation was revoked);
- MUST include Action Buttons to enable lifecycle management of the Attestation and allow the User to revoke or update it at any time (see *Management of Electronic Attestations* (page 94));
- MUST guarantee that the Attestation is a functional element for the User to be authenticated by a Relying Party in a digital context (see *Authentication* (page 69)), to access services in proximity contexts where applicable, and to request the issuance of additional EAAs where the related issuer requires person identification data (see *Issuance of Electronic Attestations of Attributes* (page 42));
- MUST display a method of assistance provided by the relevant issuer of the Attestation (see *User Assistance* (page 105));
- MUST guarantee that the Attestation is recognizable by the User and distinguishable from other EAAs.

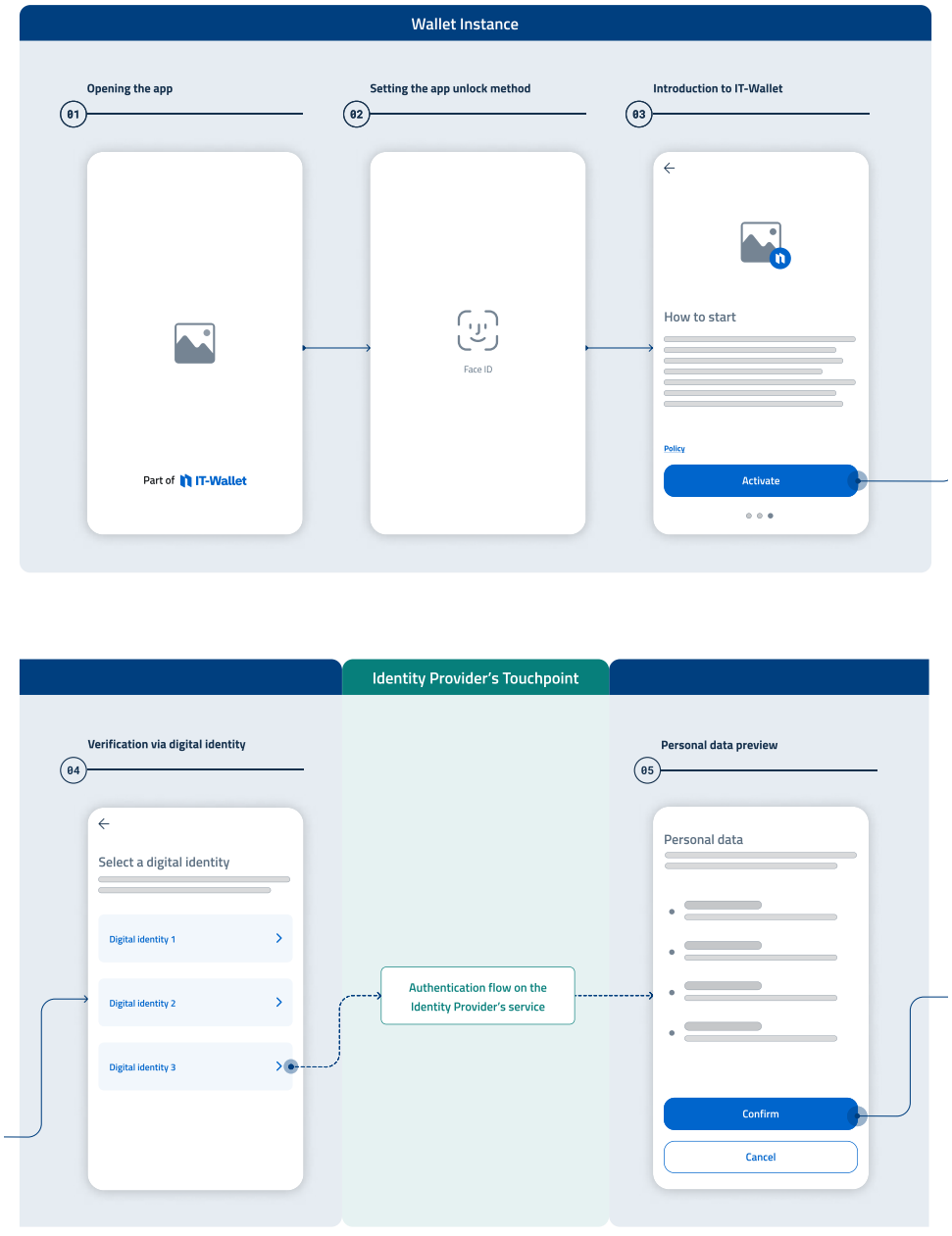


Fig. 5.2: Example of User Experience in Activating a Wallet Instance - 01



Fig. 5.3: Example of User Experience in Activating a Wallet Instance - 02

PID

The **PID** is the Person Identification Data under the European Digital Identity framework (see *PID Data Model* (page 380) and *Person Identification Data*). It is issued by the PID Provider as a technical PID (signed data structure) for presentation from the Wallet Instance.

- The Wallet Instance in the **Operational** state allows the PID issuance. Upon successful issuance of a valid PID, the Wallet Instance MUST transition to the **Valid** state (see *Transition to Valid* (page 283)).
- While the Wallet Instance is **Valid** and the PID is active/valid, the User MAY present the PID and use it for authentication and further credential issuance according to this specification and the EUDI framework.
- Revocation, expiry, or deletion of the PID MUST transition the Wallet Instance back to **Operational** (see *Transition to Operational* (page 282)).
- Where the User revokes the PID, the Wallet Provider MUST handle the consequences on the Wallet Instance lifecycle and on dependent EAAs as defined in this specification.
- The PID is intended for use also in cross-border interactions within the EUDI Wallet ecosystem, in addition to national use.

IT-Wallet ID

The **IT-Wallet ID** is the Electronic Attestation of Person Identification Data with national scope, issued for national use only (see *IT-Wallet ID Data Model* (page 392) and *IT-Wallet ID*). It is an EAA issued by an Electronic Attestation of Attributes Provider, unlike the PID.

- Issuance of the IT-Wallet ID does **not** transition the Wallet Instance to the **Valid** state; the Wallet Instance remains **Operational** (see *Transition to Operational* (page 282)).
- The IT-Wallet ID is exclusively intended for national use and not for cross-border interactions.
- To ensure a consistent identification and representation of the IT-Wallet ID across different Wallet Solutions, the Wallet Provider:
  - MUST use the official naming "IT-Wallet ID" in their Wallet Solutions;
  - MUST use the IT-Wallet ID official graphic asset available in the official-resources: Official Resources and MUST comply with the related usage specifications provided;
  - MUST use the IT-Wallet ID graphic asset in the `application/svg+xml` data format;
  - MUST NOT alter, modify, or replace the IT-Wallet ID graphic asset with unofficial graphic assets;
  - MUST maintain the minimum clear space as defined in the official-resources:Official Resources to ensure visibility and recognizability. No other graphic or textual elements MUST interfere with this space;
  - MUST NOT resize the IT-Wallet ID graphic asset below the minimum dimensions specified in the official-resources:Official Resources to maintain legibility across formats and devices;
  - MUST NOT place the IT-Wallet ID graphic asset on backgrounds that compromise its visibility or legibility. Adequate contrast between the IT-Wallet ID graphic asset and the background MUST be ensured, in line with the official-resources:Official Resources.

  Although the term IT-Wallet ID refers to the Electronic Attestation of Personal Identification Data, the IT-Wallet ID graphical asset MAY be used by the Wallet Provider to facilitate the User in identifying, more generally, their personal identification Attestations (IT-Wallet ID / PID).

The IT-Wallet ID Official Resource is provided in the related official-resources:Official Resources Section. Additional documentation on the IT-Wallet ID graphic asset is available in the Brand Manual, indicated in the official-resources:Official Resources section.



Fig. 5.4: IT-Wallet ID official graphic asset on a light background

5.4 Issuance of Electronic Attestations of Attributes

Once activation is complete, the User MAY request and obtain one or more Electronic Attestations of Attributes within their Wallet Instance.

The Electronic Attestations of Attributes is a dynamic object that enables the User to demonstrate or certify in a reliable and verifiable manner, a condition, status or right, based on the information it contains. In particular, the Electronic Attestations of Attributes:

- **is defined by an Authentic Source**, the entity holding the source data;
- **consists of a set of Attributes**, the granular data that define it (no static images or PDFs), such as, for example, a characteristic (e.g. "resident") or a status (e.g. "student");
- **is issued by a Provider of Electronic Attestation of Attributes**, the entity that emits it on the basis of the information acquired and manages its lifecycle.

Depending on the User's specific needs, the type of Electronic Attestation of Attributes, and the offerings available from the Wallet Provider, the Electronic Attestation of Attributes Provider, and the Authentic Source, the request of Electronic Attestations of Attributes can occur in two ways:

- **from the Wallet Instance Catalog**: the User explores the list of Electronic Attestations of Attributes provided by the Wallet Solution, selects the one of interest, and initiates the request process, concluding with the issuance of the Electronic Attestation of Attributes in the Wallet Instance. This pathway is available for Credential types eligible for public discovery as determined by the Supervisory body policies during the onboarding process (see *Digital Credentials Catalog* (page 179)).
- **from a Touchpoint of the Authentic Source** (or the Electronic Attestation of Attributes Provider if it coincides with the Authentic Source; (see *Credential Offer Flow* (page 452)): the User interacts with the digital service of the Authentic Source, allowing them to get a specific Electronic Attestation of Attributes in their Wallet Instance via an *Engagement Button* (page 49).

Although the methods for initiating the request are different, the issuance flows share a similar structure and process.

5.4.1 Issuance from the Wallet Instance Catalog

Below are illustrated the User Experience requirements for the issuance of an Electronic Attribute Attestation from the Catalog that the Wallet Solution Provider MUST guarantee through their own Wallet Solution:

- The User accesses their Wallet Instance using the unlock method previously set;
- The User selects the Electronic Attestation of Attributes they wish to request from the available options in the Catalog;
- The User selects from which Electronic Attestation Provider they want to obtain the Electronic Attribute Attestation, if there is more than one;
- The User views any additional information on requirements and/or limitations related to obtaining the Electronic Attestation of Attributes from the Authentic Source;
- The User views their personal identification data, if required by the Authentic Source for the request of the Electronic Attestation of Attributes, the name of the related Electronic Attestation of Attributes Provider, and any related information policy. The User gives their consent to proceed, presenting the required data to the Electronic Attestation of Attributes Provider, or cancels the operation;
- The User views a preview of the Electronic Attestation of Attributes. The User confirms the data shown in the preview to proceed with the request or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User views the positive outcome of the request;
- The User views the details of the requested Electronic Attestation of Attributes, including: the data contained in it, the name of the Electronic Attestation of Attributes Provider who issued the Attestation, and the name of the Authentic Source;
- The User has access to all issued Electronic Attestations by navigating the Wallet Instance.

The Authentic Source MAY provide additional information related to an Electronic Attestation of Attributes. This information MUST be displayed by the Wallet Instance to the User, before initiating the process of issuing the Electronic Attestation of Attributes. In order to properly draft this informational content, the Authentic Source:

- MUST use clear language (e.g. avoid technical or complex terms), be concise (e.g. avoid excessively long or elaborate texts) and inclusive (e.g. avoid ability-based verbs), following the best practices for writing, language and tone of voice described in [REF_ACCESSIBILITY] and, in the case of public entities, in [GL_DESIGN];

- MUST adhere to the specific purpose of the text, communicating useful information to the User before engaging in the issuance process (e.g. listing prerequisites or stating limitations that could affect the successful outcome of the procedure);
- MUST ensure that the information is constantly updated;
- MUST include a title and text in which it MAY include references to external channels to direct Users to a procedure, explore a specific topic and/or open support requests.

Following is an example of informative text:

Note

Title: Who may obtain the document **Text:** The digital version of [Document name] is available only to those who already hold the physical one. Please, make sure to have already obtained the corresponding physical document. For more details, [read more information] (URL).

For further information, please refer to the section *Authentic Source Registry* (page 163) (see `data_capabilities.user_information` parameter).

The Wallet Provider MUST allow the User to remove an Electronic Attestation of Attributes through their Wallet Instance at any moment. In case of absence of the device where the Wallet Instance was activated, the Wallet Provider MUST allow the User to deactivate the entire Wallet Instance through a specific Touchpoint. In addition, the Electronic Attestation of Attributes Providers SHOULD allow the User to revoke the issued Digital Credentials through specific Touchpoints. For more details, please refer to the *Deactivation of the Wallet Instance* (page 101) and *Management of Electronic Attestations* (page 94) sections.

In the event of communication issues between the systems of the Electronic Attestation of Attributes Provider and the Authentic Source, or if administrative or technical processes prevent the immediate issuance of the Electronic Attestation of Attributes, the actors involved MAY support a deferred issuance process. In this case the Wallet Provider MUST guarantee that:

- Upon reaching the final step of the process, the User visualizes a message prompting them to wait until the Electronic Attestation of Attributes can be issued.
- The User is informed by the Electronic Attestation of Attributes Provider once the Electronic Attestation of Attributes becomes available.

If the User encounters incorrect data in an already obtained or in-progress Electronic Attestation of Attributes, the Wallet Provider SHOULD guarantee the User appropriate assistance via their Wallet Instance. For more information, please refer to the *User Assistance* (page 105) section.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

5.4.2 Issuance from a Touchpoint of the Authentic Source

Below are illustrated the User Experience requirements for the issuance of an Electronic Attribute Attestation from a Touchpoint of the Authentic Source (also known as *Credential Offer Flow* (page 452)) that the Authentic Source MUST guarantee through their Touchpoint:

- The User interacts with the *Engagement Button* (page 49) clearly displayed in the Touchpoint interface;
- The User selects the Wallet Solution with which to proceed, through an interface that MUST follow the directions and functionalities described for the *Selection Page* in the *Authentication* (page 69) section;
- (*cross-device only*) the User scans the QR code that invokes the opening of their chosen Wallet Instance, through an interface that MUST follow the directions and functionalities described for the *QR Code Page* in the *Authentication* (page 69) section; alternatively the User can exit the flow.
- The User accesses their Wallet Instance using the unlock method previously set;
- The User views their personal identification data, if required for the request of the Electronic Attestation of Attributes, the name of the related Electronic Attestation of Attributes Provider, and any related information policy. The User gives their consent to proceed, presenting the required data to the Electronic Attestation of Attributes Provider, or cancels the operation;
- The User views any additional information on requirements and/or limitations related to obtaining the Electronic Attestation of Attributes;
- The User views a preview of the Electronic Attestation of Attributes. The User confirms the data shown in the preview to proceed with the request or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User views the positive outcome of the request;
- The User views the details of the requested Electronic Attestation of Attributes, including: the data contained in it, the name of the Electronic Attestation of Attributes Provider who issued the Attestation, and the names of the Authentic Sources.

In case of errors during the issuance of the Electronic Attestation of Attributes, the Authentic Source MUST guarantee that the User receives consistent messages that inform them and guide



Fig. 5.5: Example of User Experience in the issuance an Electronic Attestation of Attributes from Catalog - 01



Fig. 5.6: Example of User Experience in the issuance an Electronic Attestation of Attributes from Catalog - 02

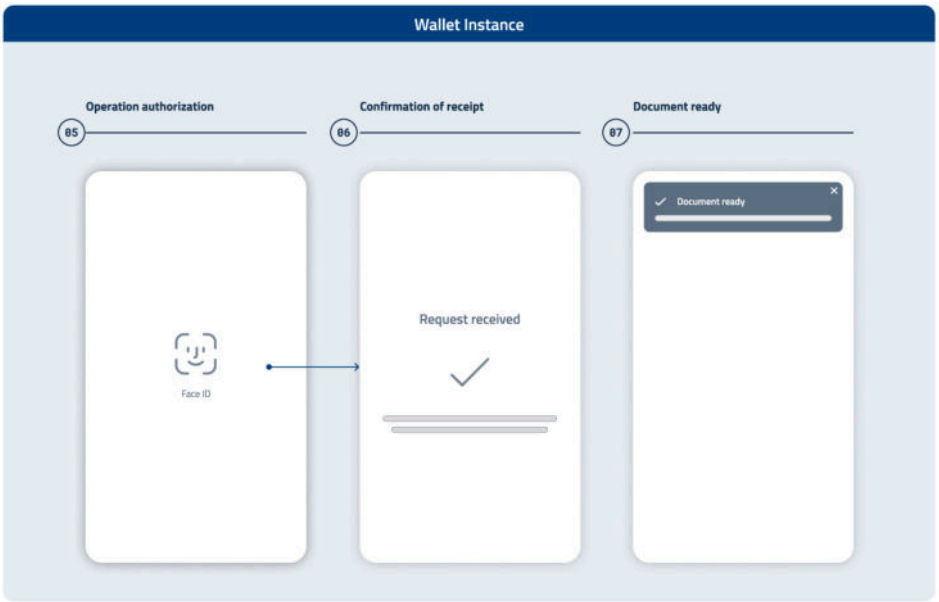


Fig. 5.7: Example of User Experience in the issuance an Electronic Attestation of Attributes from Catalog - 03

them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

Following is represented for illustrative purposes the page with the *Engagement Button* (page 49), together with the Selection Page and the QR Code Page with the interface elements and texts updated according to the *Issuance from a Touchpoint of the Authentic Source* (page 45) flow.

Engagement Button

The Engagement Button is an interactive element of the interface that allows the USer to start a process, for example of Authentication (”Login with IT-Wallet”), of issuance of an Electronic Attestation (”Add to It-Wallet”) or of remote presentation (”Verify with It-Wallet).

Relying Parties MAY make available the Engagement Button in their Touchpoint’s page to allow the User to use their services through a Wallet Instance.

The Engagement Button:

- MUST be used exactly as outlined in the Official Resources and MUST NOT be redesigned ad hoc;
- MUST be used only in the shapes, colors and proportions defined and MUST NOT be altered, distorted, or hidden;
- MUST be responsive to all screen resolutions and MUST be integrated in the Discovery Page in order to meet minimum usability and accessibility requirements;
- MUST maintain a minimum distance from other elements (quiet zone) of at least 24px;
- MUST state in their label the action to be done followed by ”IT-Wallet”, for example ”Login with IT-Wallet”, ”Add to IT-Wallet”, ”Verify with IT-Wallet”;
- MAY be accompanied by text explaining the added value of performing that action using the IT-Wallet system.
- Actors wishing to integrate the Engagement Button into their Touchpoints MUST ensure that it is translated into other languages, at least English.

Below are some non-mandatory examples of Engagement Button labels: - Login with IT-Wallet (see *Authentication Button* (page 90))

- Add to IT-Wallet (see *Issuance from a Touchpoint of the Authentic Source* (page 45))
- Verify with IT-wallet (see *Remote Presentation* (page 65))

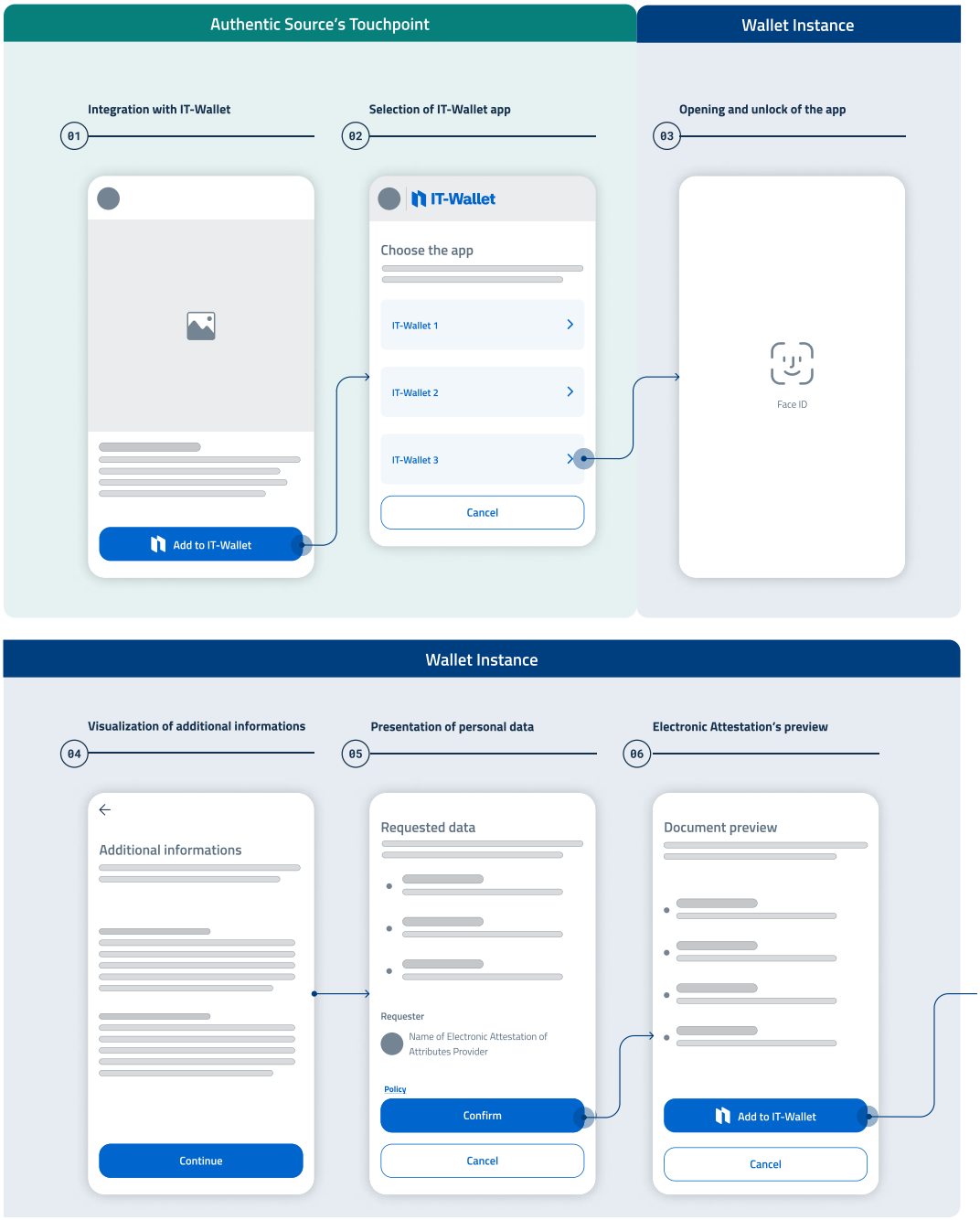


Fig. 5.8: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, same device - 01

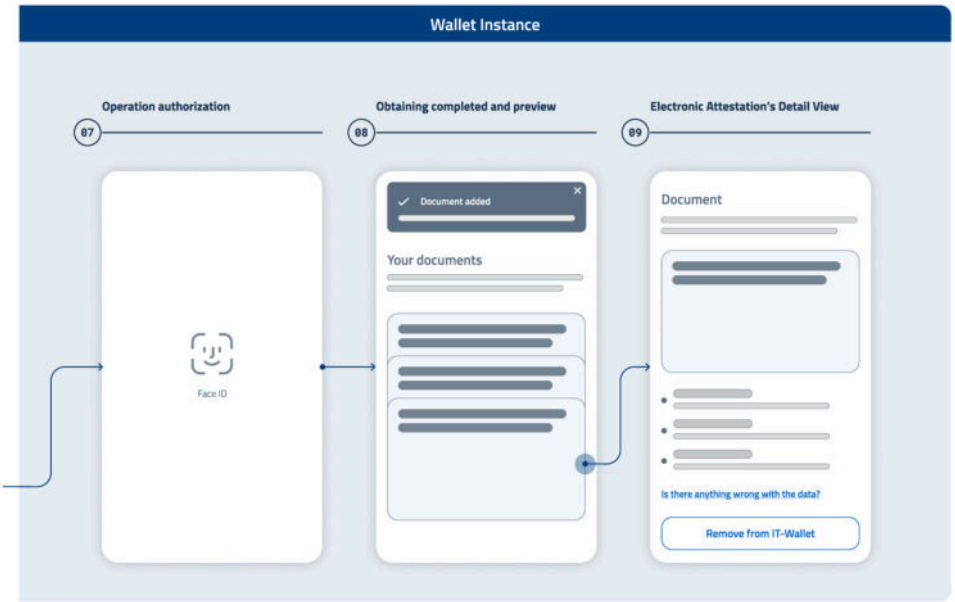


Fig. 5.9: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, same device - 02

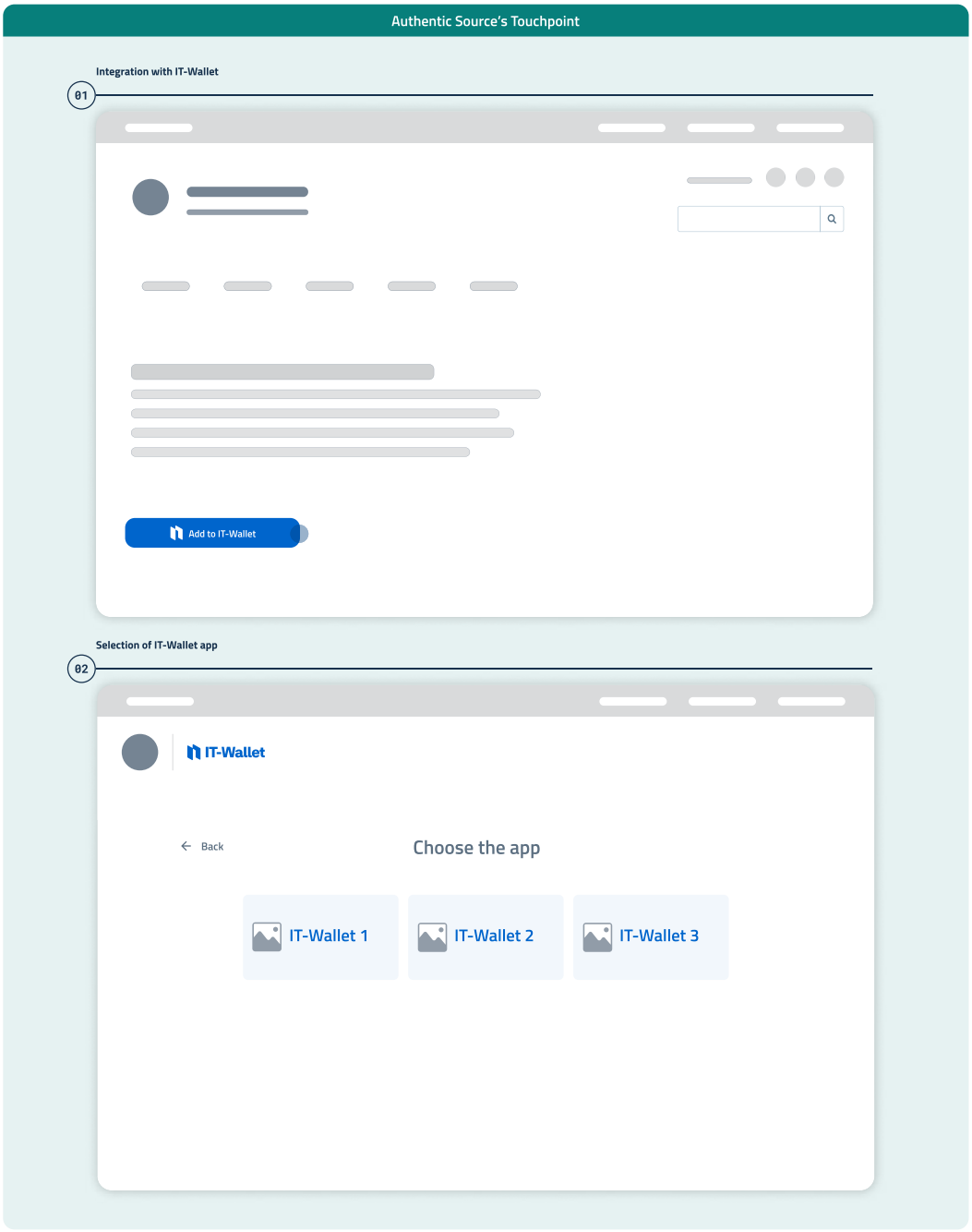


Fig. 5.10: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, cross device - 01

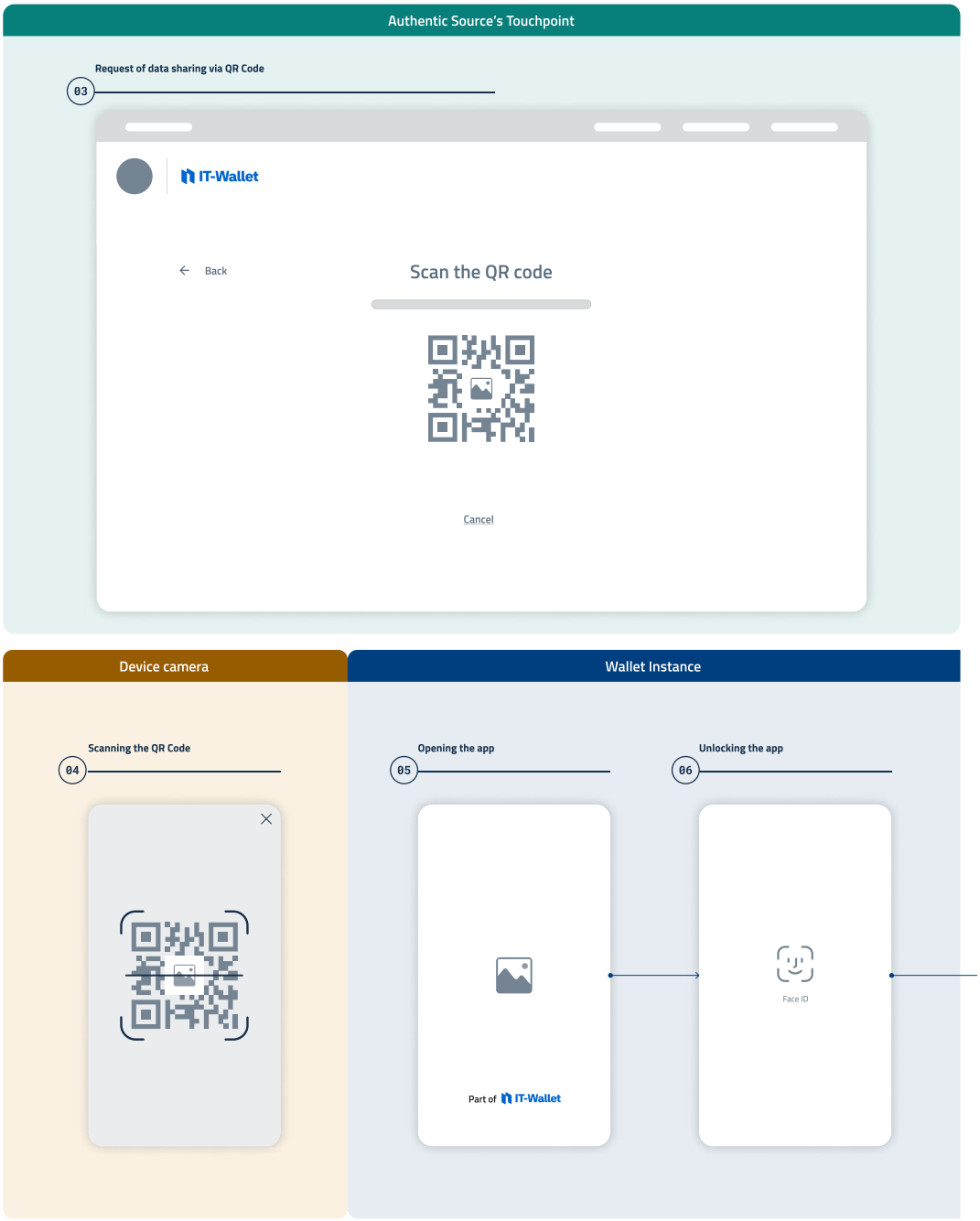


Fig. 5.11: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, cross device - 02



Fig. 5.12: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, cross device - 03

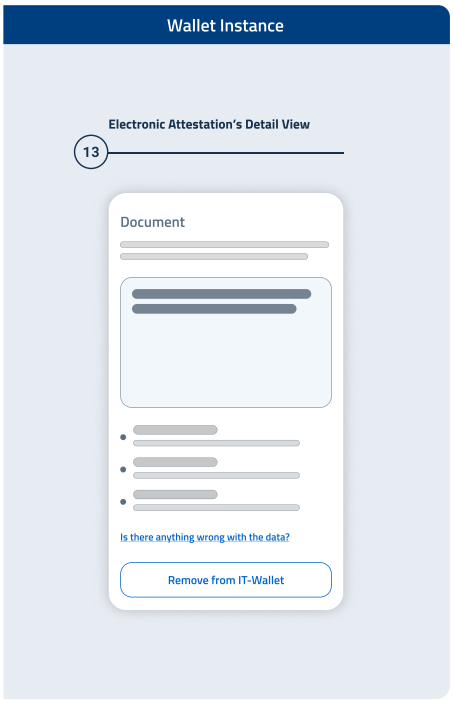


Fig. 5.13: Example of User Experience in the issuance of an Electronic Attestation of Attributes from the Authentic Source, cross device - 04

5.4.3 Focus on Electronic Attestations of Attributes

Designing and implementing an Electronic Attestation of Attributes properly is essential to bring real value to the entire system. In this regard, the role of the Authentic Source and the Wallet Provider is crucial.

- The Authentic Source is responsible for designing all the core features of the Electronic Attestation of Attributes in terms of:
- **opportunity and perceived value**: what the EAA can attest, in which contexts of use and by whom;
- **ease of understanding**: how simple and intuitive it is for those who use it and for those who verify it;
- **communicative effectiveness**: how recognizable or reliable it is perceived to be by those who use or verify it.
- The Wallet Provider is responsible for presenting the Electronic Attestation of Attributes in accordance with:
- the provisions set out in these Technical Specifications, with a view to ensuring system consistency and adherence to the principles of usability [GL_DESIGN] and accessibility [REF_ACCESSIBILITY];

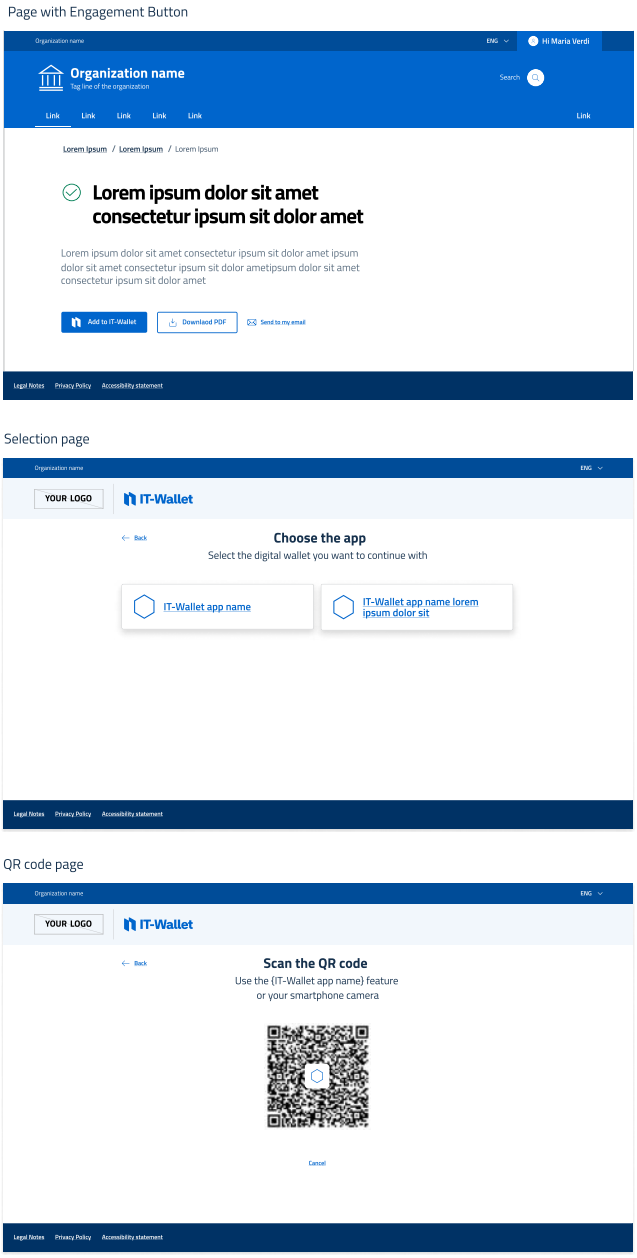


Fig. 5.14: Illustrative page with Engagement Button, Selection page and QR code page for the credential offer

- the provisions of the Authentic Sources Registry and the Digital Credentials Catalog (see *Registry Infrastructure* (page 149));
- the UX/UI design of their Wallet Solution.

In order to guide the Authentic Source in designing and the Wallet Provider in accurately representing an EAA, below are the elements to be considered and the structure to be adopted at the Detail View level:

- **EAA name**, which is the official, distinctive and human-readable name of an EAA;
- **Identification Attributes** (if available), the User's basic personal details in the following order:
- Given name
- Last name
- Tax code
- Date of birth
- Place of birth
- Etc.
- **Level I Attributes**, which are the data that represent and identify the specific type of EAA (e.g. driving licence number, category, expiry date, etc.)
- **Level II Attributes** (if available), which are detailed data used to describe or elaborate on a specific Level I Attribute. These Level II Attributes may be structured as:
- **List of descriptions**, which is a list of values relating to the same object;
- **List of Attributes with value**, which is a list of additional objects with associated values;
- **Metadata**, which are data identifying the details of the EAA at the level of:
- **Authentic Source name**, which is the identifying name of the data owner or the official database holding the information behind the EAA, defined by the parameter `data_capabilities.data_origin`;
- **Provider of Electronic Attestation of Attributes name**, which is the identifying name of the entity that creates and issues the EAA, defined by the parameter `organization_name`.

Below are the detailed requirements for the design and the representation of EAA within the Wallet Solutions (see *Authentic Source Registry* (page 163) and *Digital Credentials Catalog* (page 179)). In particular:

The Authentic Source:

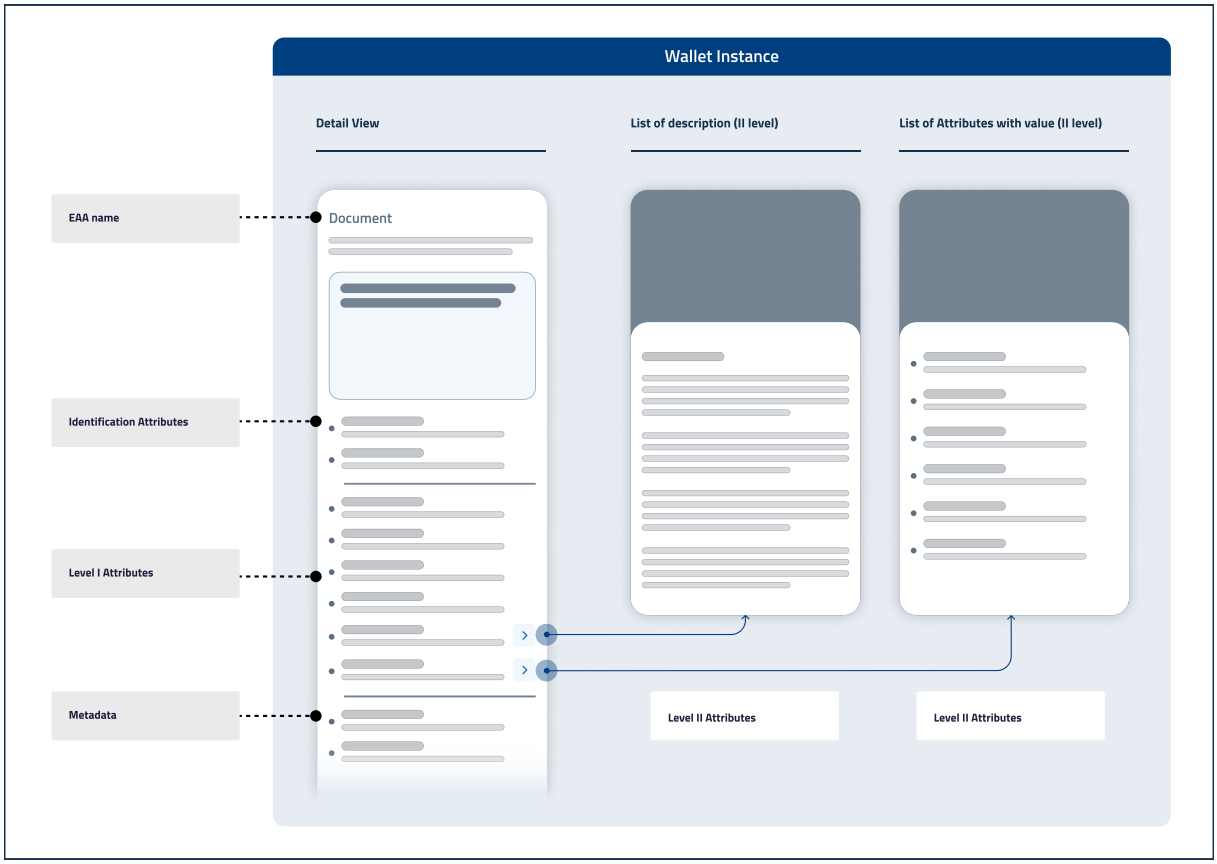


Fig. 5.15: Example of Electronic Attestation of Attributes layout, Detail View.

- MUST specify the **EAA name** using the `credential_name` parameter. In particular, the Authentic Source:
  - MUST use terms or phrases that correspond to or are similar to the name of any pre-existing version of the document to which the EAA corresponds, so as to facilitate recognition by the User;
  - MUST use clear and simple terms, avoiding anglicisms, acronyms or technical terms where possible;
  - SHOULD define a short and concise name.
- MUST specify the data owner or data source name using the parameter `data_capabilities.data_origin`. In particular, the Authentic Source:
  - MUST use the official name to ensure maximum recognisability and trust by the User;
  - MUST follow the structure "[Acronym] - [Name]" if it wishes to use an acronym alongside the full name (e.g. "ANIS - National Register of higher education");
  - MAY follow the structure "[Primary entity acronym] - [Secondary entity name]" if it is useful to separate the acronym from the full name in order to add a further level of detail and enhance the recognisability and sense of reliability of the entity in charge (e.g. "MIT - General Directorate for Motor Vehicles").
- MAY provide the **Authentic Source's logo** in two versions, a compact version via the `organization_info.logo_uri` parameter and an extended version via the `organization_info.logo_extended_uri` parameter. In particular, the Authentic Source:
  - MUST provide the logo only in one of the following formats: `image/png`, `image/svg+xml`, or `image/webp`;
  - MUST provide the logo in both a positive and a negative version, if available;
  - MUST provide the logo with a minimum size of 30 × 30 pixels and a maximum size of 60 x 60 pixels, in its compact version;
  - MUST provide the logo with an aspect ratio of 1:1, in its compact version;
  - MUST provide the logo which does not exceed the maximum file size of 80 KB, in its compact version;
  - MUST provide the logo with a minimum size of 200 × 30 pixels and a maximum size of 650 × 180 pixels, in its extended version;
  - MUST provide the logo which does not exceed the maximum file size of 150 KB, in its extended version.
- MAY provide a distinctive **EAA logo** via the `data_capabilities.logo_uri` parameter. In particular, the Authentic Source:

  - MUST provide the logo only in one of the following formats: `image/png`, `image/svg+xml`, or `image/webp`;
  - MUST provide the logo in both a positive and a negative version, if available;
  - MUST provide the logo with a minimum size of 200 × 30 pixels and a maximum size of 650 × 180 pixels;
  - MUST provide the logo with a maximum file size of 150 KB.

- MAY define a distinctive **EAA color** to be associated with a specific EAA via the `data_capabilities.background_color` parameter. The aim is to characterise the EAA, making it recognisable to the User, particularly in cases where the intention is to give a new EAA a strong identity or to maintain consistency with a corresponding physical document (e. g., consider the italian driving licence and its distinctive pink colour). In particular, the Authentic Source:
  - MUST provide the background color only using one of the following color modes: HEX, HSB, RGB, sRGB, HSL, or HSV.
- MUST specify the desired **Attributes order** via the `data_capabilities.available_claims.order` parameter. In particular, the Authentic Source:
  - MUST order the data according to the default structure:
    * Identification Attributes (optional);
    * Level I Attributes;
    * Level II Attributes (optional);
  - MUST NOT exceed two levels of nesting in the organisation of attributes;
  - SHOULD NOT exceed a maximum of 15 Attributes per level;
  - SHOULD NOT exceed a maximum of 85 characters for Attribute values;
  - SHOULD NOT exceed a maximum of 181 characters for the value of each description in the case of Level II Attributes with a "list of descriptions".

The Wallet Provider:

- MUST clearly and accessibly display the identifying **EAA name** as defined by the `credential_name` parameter within the *Digital Credentials Catalog* (page 179);
- MUST clearly and accessibly display the **EAA Attributes**, respecting the predefined macro-structure of the Detail View, the division of Attributes into level I and level II (where available), and the data ordering as defined by the parameter `data_capabilities.available_claims_order` within the *Authentic Source Registry* (page 163). In particular, the Wallet Provider:
  - MUST represent the EAA in accordance with the predefined structure:

    * EAA name;
    * Identification Attributes (optional);
    * Level I Attributes;
    * Level II Attributes (optional);
    * Metadata.
  - MUST include the **Authentic Source's logo** and/or the specific **EAA logo** if provided by the Authentic Source through the *Authentic Source Registry* (page 163);
  - MUST ensure the adoption of the **EAA color** defined by the Authentic Source, if specified within the *Authentic Source Registry* (page 163), at the hue (H) level. The Wallet Provider MAY optimize saturation (S) and brightness (B) values to adapt the specific color to accessibility requirements and/or to the graphical design choices of its Wallet Solution. If the Authentic Source does not provide color specifications, the Wallet Provider is required to define and adopt its own default graphical choices.

The Wallet Provider is the ultimate responsible party for the visual presentation of the Electronic Attestation of Attributes within its Wallet Solutions. Therefore, to ensure a high level of accessibility [REF_ACCESSIBILITY] and usability [GL_DESIGN], further user experience requirements are set out below. In particular, the Wallet Provider:

- MUST correctly display the EAA across all devices, ensuring a consistent experience on screens of varying sizes;
- MAY display the EAAs issued in the form of cards stacked in a list in the Preview View, in line with approaches already used by other digital wallets in the market;
- MUST optimize the EAA layout in the Preview View for scalability and usability, especially when multiple EAAs are displayed on the same screen;
- MUST display clearly the name of the EAA, as defined within the Digital Credentials Catalog via the `credential_name` parameter (see *Digital Credentials Catalog* (page 179)), in both the Detail View and the Preview View;
- MUST display the EAA status, if different from valid to provide transparency on its lifecycle and MAY display it if valid, in both the Detail View and the Preview View. Specific details about the EAA status, if invalid, MAY be provided in the Detail View (e. g., the reason why the EAA is revoked);
- MAY include optional information to enhance the User Experience and the EAA recognizability, both in the Preview View and the Detail View, such as the logo and/or the color, as defined by the Authentic Source within the Authentic Source Registry (see *Authentic Source Registry* (page 163));

- MUST include the same data of the Preview View in the Detail View and give a complete representation of all the other Attributes, following the order defined by the Authentic Source within the Authentic Source Registry (see *Authentic Source Registry* (page 163)); MAY include specific details in the Detail View, for example information about use case scenario or the reason for the invalid state of the EAA;
- MUST include Action Buttons in the Detail View to enable the EAA lifecycle management and allow the User to revoke or to update a EAA at any time (see *Management of Electronic Attestations* (page 94));
- MUST guarantee that the EAA is a functional element, for the User to access services provided by Relying Parties in digital and proximity contexts (see *Presentation of Electronic Attestations* (page 62));
- MUST display in the Detail View a method of assistance given by the Authentic Source via the `data_capabilities.contacts` parameter (see *User Assistance* (page 105) and see *Authentic Source Registry* (page 163)).

Below an example of Electronic Attestation of Attribute layout, in a Preview View, and Detail View.

5.5 Presentation of Electronic Attestations

The presentation process allows the User to access a service or demonstrate ownership of certain data or their eligibility to perform a specific action. The presentation of Electronic Attestations and their subsequent verification involves interaction between the Wallet Instance, managed by the User, and a Relying Party Instance. Depending on the circumstances and context of the interaction, the following scenarios can be outlined:

- **Proximity Presentation**: the User presents their personal identification data and/or EAA data through the Wallet Instance, directly to a Verifier or to a device designated for in-person verification.
- **Remote Presentation**: the User presents their personal identification data and/or EAA data through the Wallet Instance, to a Relying Party configured for online verification, for instance, to Authenticate and access the services offered.

Regardless of the type of presentation, the Wallet Provider MUST allow the Wallet Instance to inform the User of the identity of the Relying Party. In case of a Relying Party Intermediary, the User MUST be informed of the intermediary's involvement during the presentation request phase. The identity of the primary Relying Party MUST always be visible to the User and MUST NOT be replaced by that of the Relying Party Intermediary.

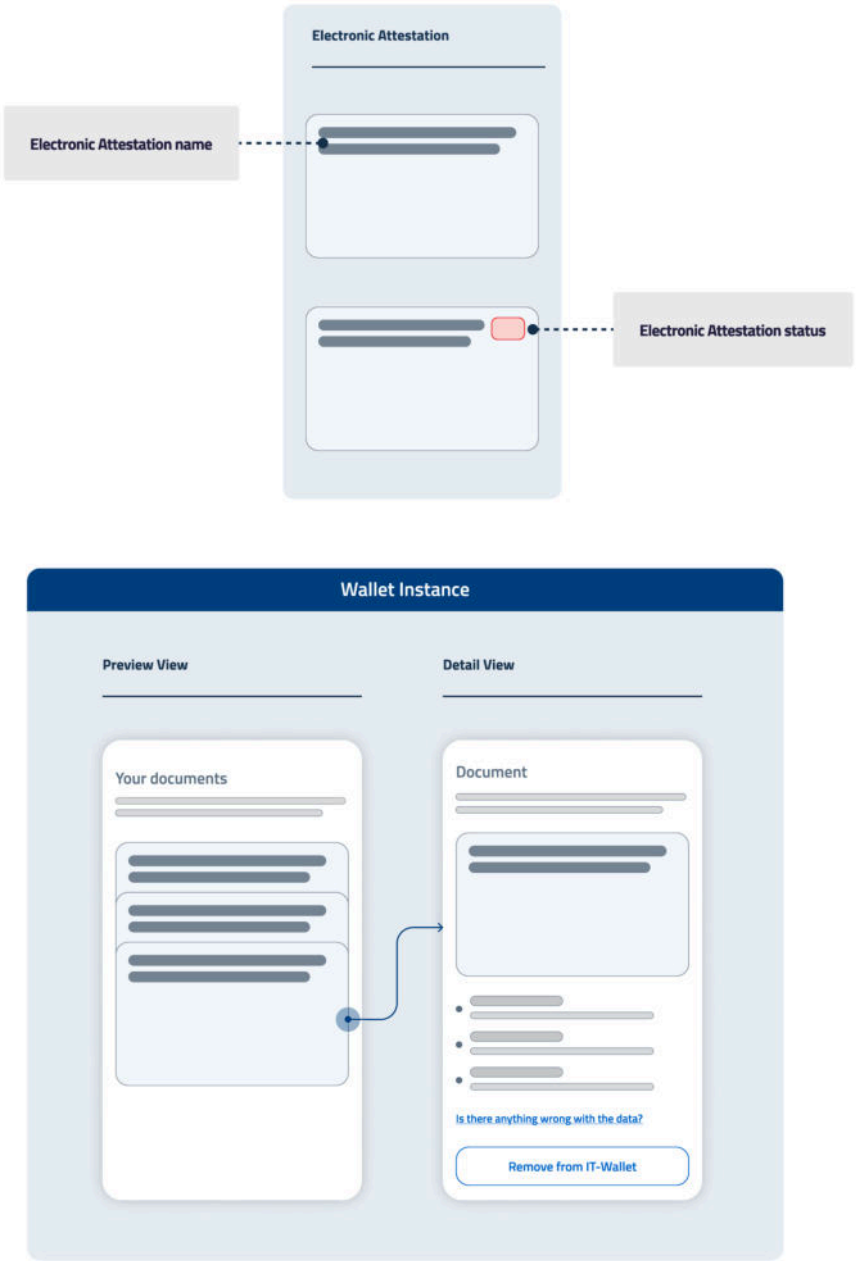


Fig. 5.16: Example of Electronic Attestation of Attribute layout, Preview View, and Detail View.

5.5.1 Proximity Presentation

Proximity presentation allows the User to present their personal identification data and/or EAA data via their Wallet Instance, using one of two methods:

- **Supervised mode**: the User presents their personal identification data and/or EAA data through the Wallet Instance to a Verifier (e.g., law enforcement officer, desk operator) equipped with a dedicated verification system (*Mobile Relying Party Instance* (page 343)).
- **Unsupervised mode**: the User presents their personal identification data and/or EAA data through the Wallet Instance to a designated device (e.g., turnstile, totem) provided with a dedicated verification system (Embedded Relying Party Instance).

Below are the User Experience requirements related to both methods that the Wallet Provider MUST guarantee via their Wallet Solution.

Supervised Mode

- The User accesses their Wallet Instance using the unlock method previously set;
- The User navigates to the feature dedicated to QR Code generation;
- The User presents the generated QR Code to the Verifier acting on behalf of the Relying Party, who scans it using the designated verification app or system;
- The User reviews their requested personal identification data and/or EAA data, the name of the requesting Service Provider, and any related policy. The User decides whether to present any non-mandatory data (Selective Disclosure). The User provides consent to proceed or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User receives confirmation of the successful presentation.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

Unsupervised Mode

- The User accesses their Wallet Instance using the unlock method previously set;
- The User navigates to the feature dedicated to QR Code generation;
- The User presents the generated QR Code to the designated device (e.g., a turnstile) of the Relying Party for scanning;
- The User reviews their requested personal identification data and/or EAA data, the name of the requesting Service Provider, and any related policy. The User decides whether to present any non-mandatory data (Selective Disclosure). The User provides consent to proceed or cancels the operation;

- The User authorizes the operation using the unlock method previously set;
- The User receives confirmation of the successful presentation.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

5.5.2 Remote Presentation

Remote presentation allows the User to present their personal identification data and/or EAA data by interacting with a Relying Party's Touchpoint through the designated *Engagement Button* (page 49).

This presentation can occur in two different modes, depending on the type of device used to access the service:

- **Same-device mode**: when the User accesses an online digital service integrated with a special verification system (*Web Relying Party Instance* (page 351)) using the same device which the Wallet Instance is installed on;
- **Cross-device mode**: when the User accesses a digital service integrated with a special verification system (*Web Relying Party Instance* (page 351)) using a different device from the one which the Wallet Instance is installed on.

Below are the User Experience requirements related to both methods that the Wallet Provider MUST guarantee via their Wallet Solution.

Same-Device Mode

- The User clicks the *Engagement Button* (page 49) provided on the Relying Party's Touchpoint;
- The User selects the Wallet Solution to proceed with, through an interface that MUST follow the instructions and functionalities provided for the Selection Page described in the *Authentication* (page 69) section;
- The User accesses their Wallet Instance using the unlock method previously set;
- The User reviews the requested personal identification data and/or EAA data, the name of the requesting Relying Party, and any related policy. The User decides whether to present any non-mandatory data (Selective Disclosure). The User provides consent to proceed or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User receives confirmation of the successful presentation within the Wallet Instance;



Fig. 5.17: Example of User Experience in proximity presentation.

- The User returns to the Relying Party's Touchpoint, where they see confirmation of the completed presentation.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

Cross-Device Mode

- The User clicks the *Engagement Button* (page 49) provided on the Touchpoint of the Relying Party while accessing the service from a different device than the one where the Wallet Instance is installed;
- The User selects the Wallet Solution to proceed with, through an interface that MUST follow the instructions and functionalities provided for the Selection Page described in the *Authentication* (page 69) section;
- The User scans the QR Code provided by the Relying Party using their Wallet Instance or the camera on their device; the QR code interface MUST follow the instructions and functionalities provided for the QR Code Page described in the *Authentication* (page 69) section;
- The User reviews the requested personal identification data and/or EAA data, the name of the requesting Relying Party, and any related policy. The User decides whether to present any non-mandatory data (Selective Disclosure). The User provides consent to proceed or cancels the operation;
- The User authorizes the operation using the unlock method previously set;
- The User receives confirmation of the successful presentation within the Wallet Instance;
- The User returns to the Relying Party's Touchpoint and views confirmation of the completed presentation.

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For further details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

Following is represented for illustrative purposes the page with the *Engagement Button* (page 49), together with the Selection Page, the QR Code Page and the Thank You Page with the interface elements and texts updated according to the *Remote Presentation* (page 65) flow.

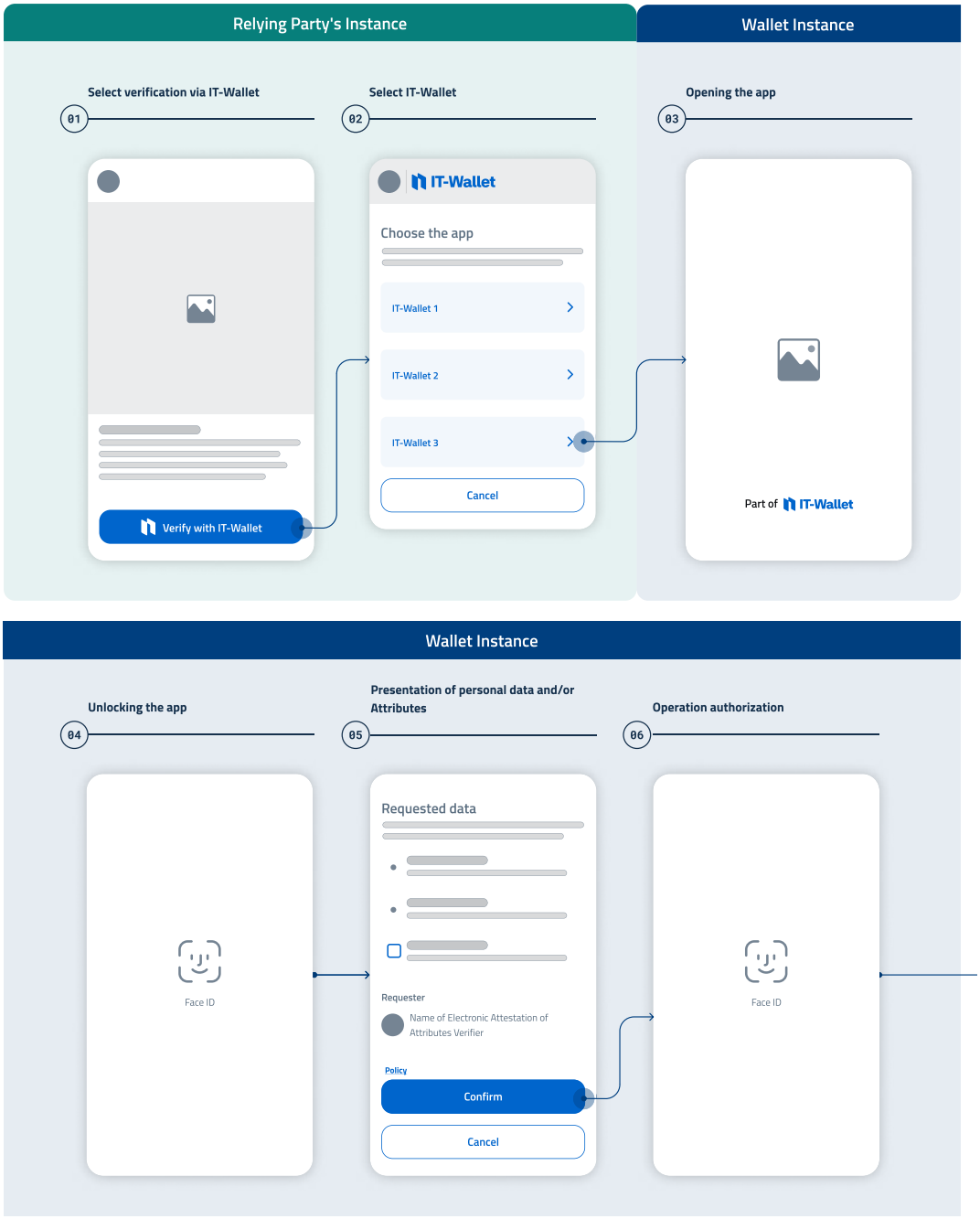


Fig. 5.18: Example of User Experience in remote, same-device presentation - 01

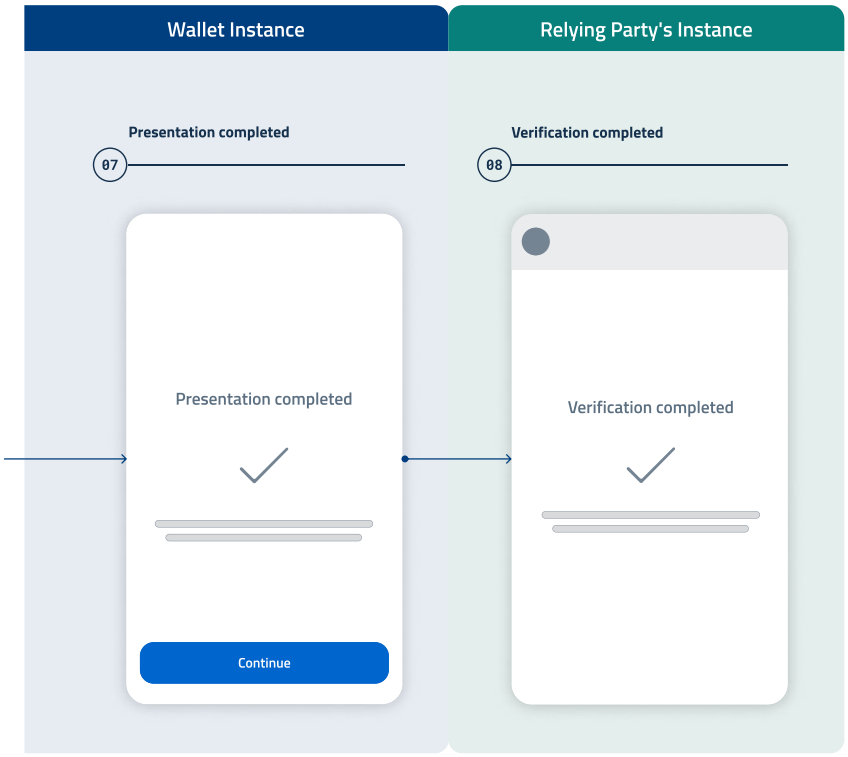


Fig. 5.19: Example of User Experience in remote, same-device presentation - 02

Authentication

Authentication is a specific use case of remote presentation that allows the User to securely access services provided by both public and private Relying Parties. This is achieved by presenting their personal identification data and, if necessary, a set of Attributes contained in the obtained Electronic Attestations of Attributes. This process ensures that the User retains control over their data, including the ability to present only the information strictly necessary for verification by Relying Parties.

The Authentication process can be carried out using both the same-device and cross-device modes described above. For the User Experience functional requirements that MUST be addressed, please refer to the functional requirements for *remote presentation* (page 65) in same-device and cross-device modes.

From a User Experience perspective, the Authentication process differs from the Presentation process only in how it is initiated, which is through a dedicated *Authentication Button* (page 90).

To ensure a consistent and seamless Authentication process across all Relying Parties, each Relying Party MUST follow the visual and User Experience requirements outlined below, together with compliance with [REF_ACCESSIBILITY] and, in the case of public entities, with [GL_DESIGN].

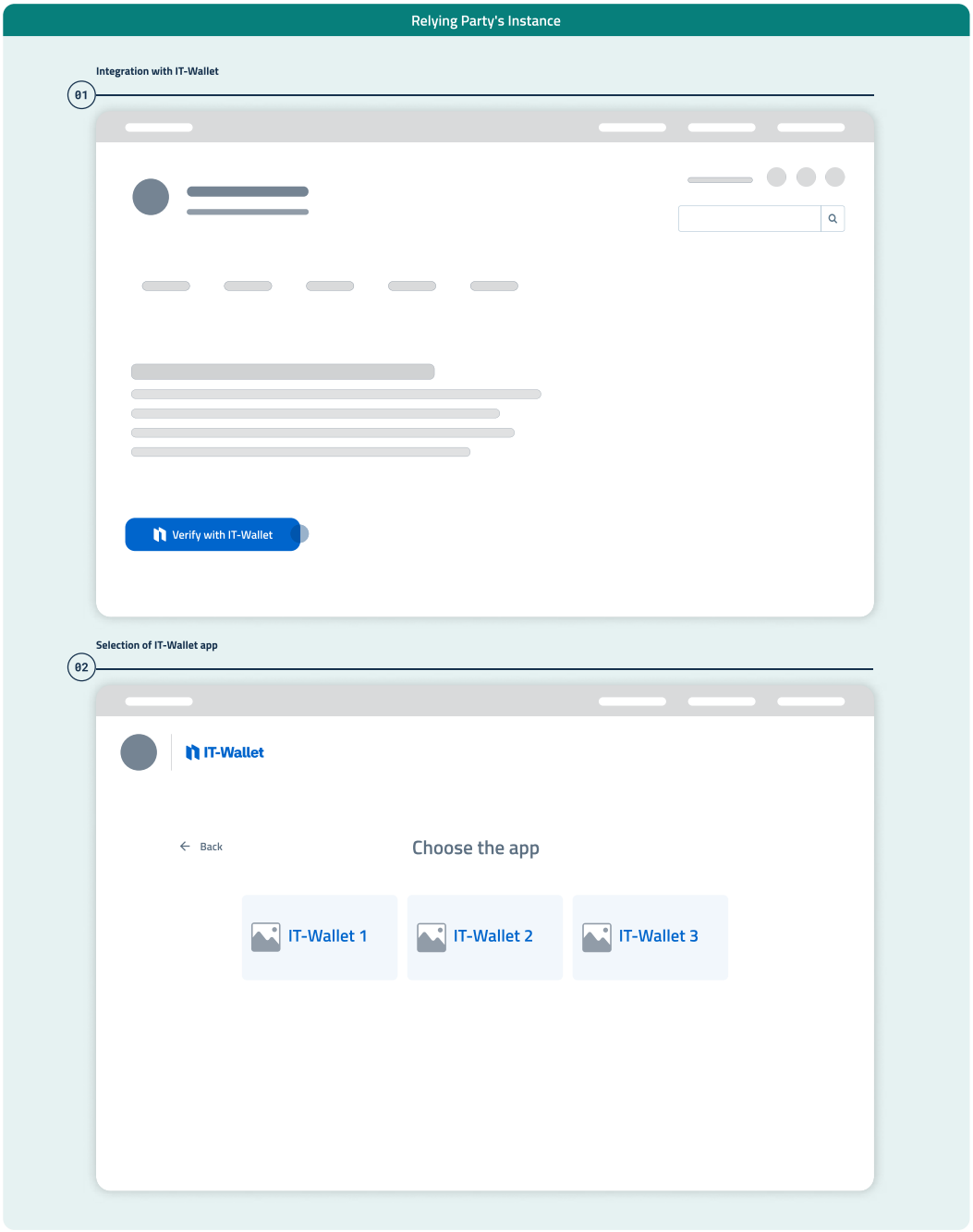


Fig. 5.20: Example of User Experience in remote, cross-device presentation - 01

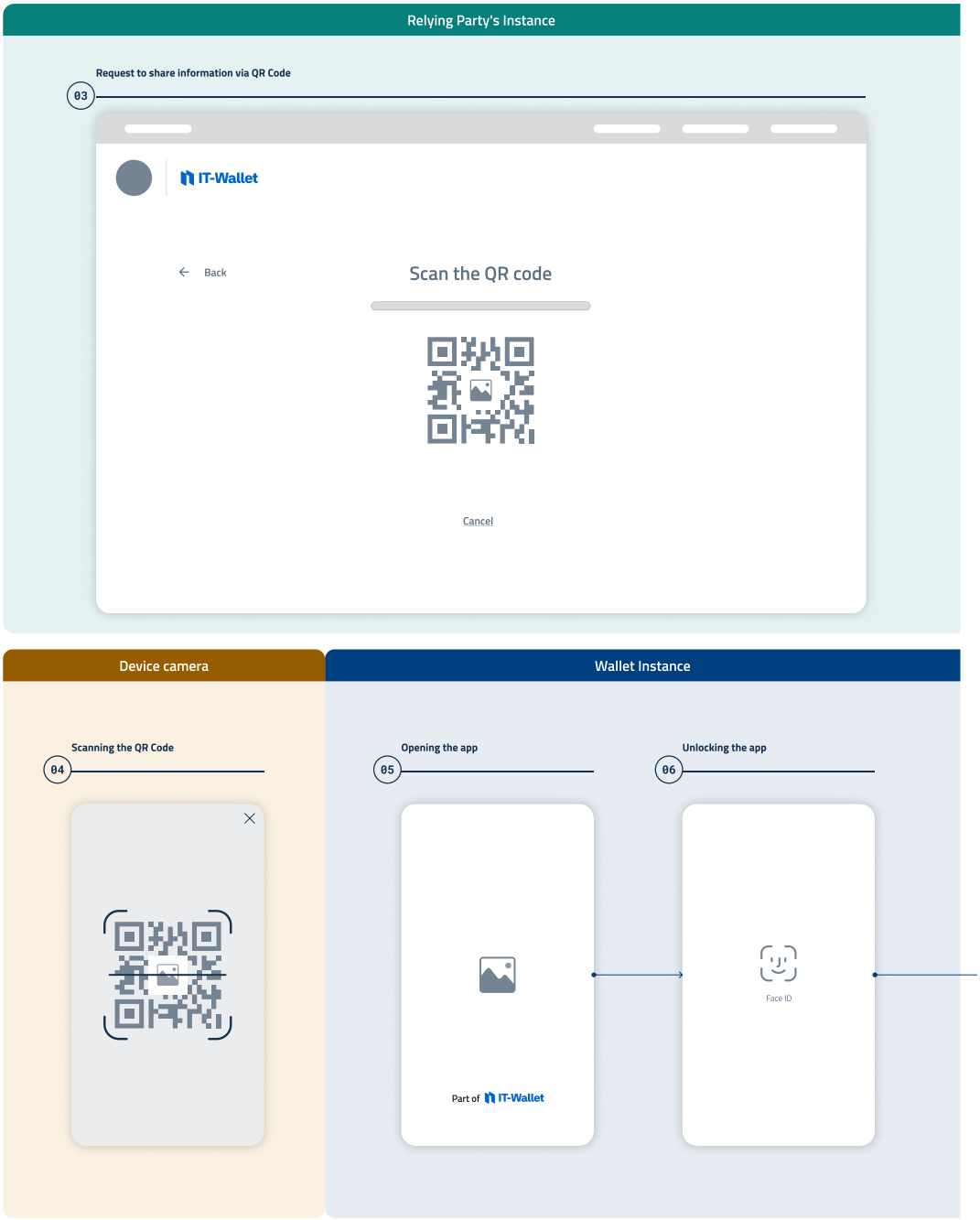


Fig. 5.21: Example of User Experience in remote, cross-device presentation - 02

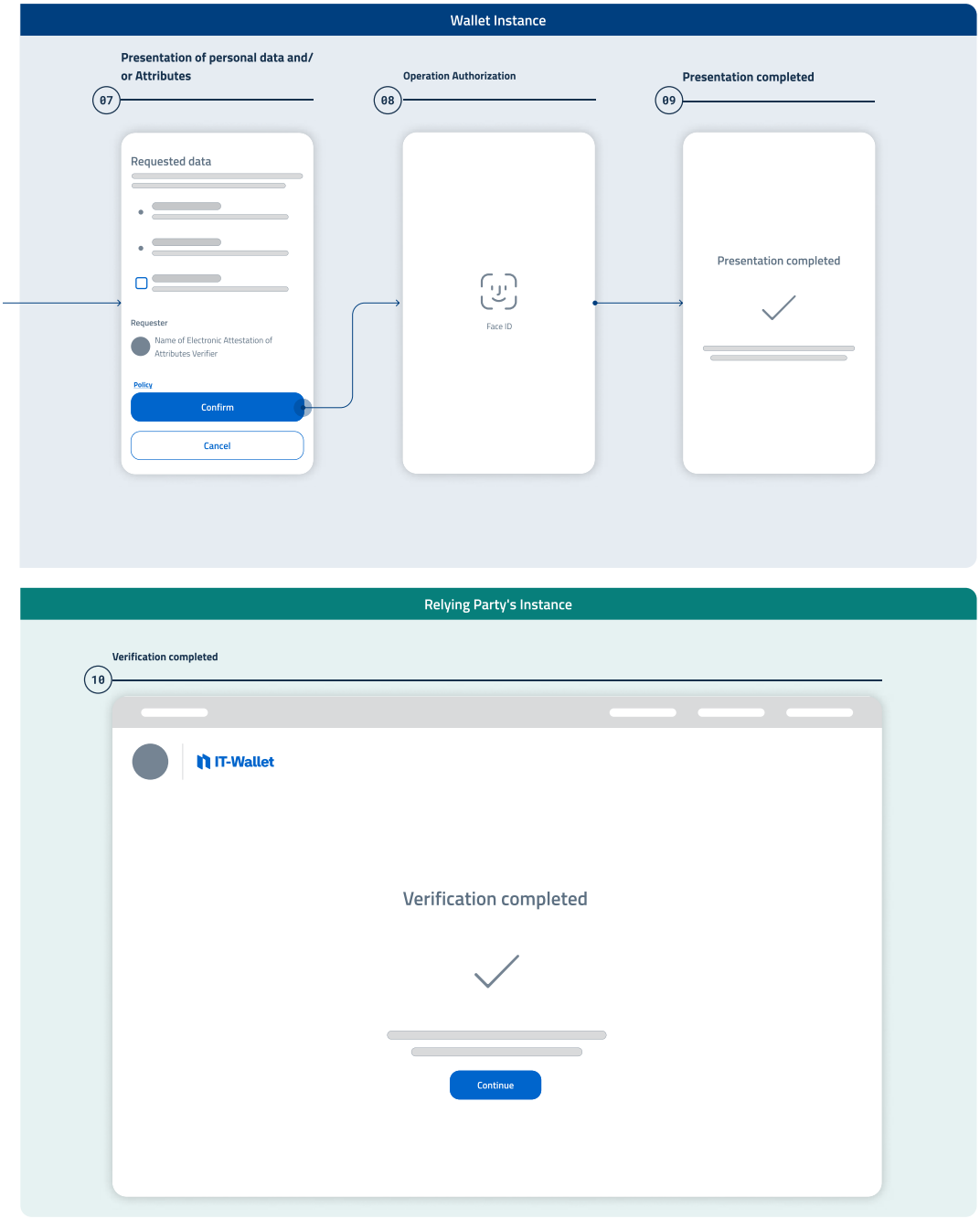
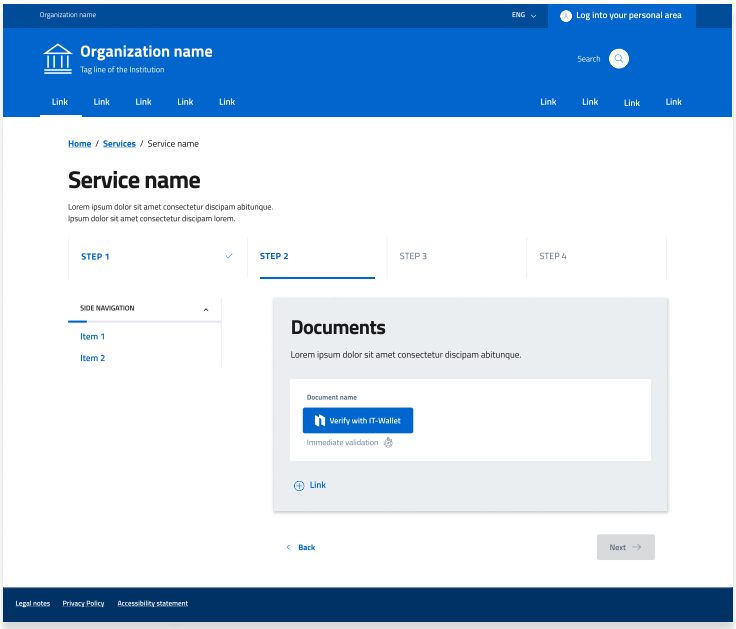


Fig. 5.22: Example of User Experience in remote, cross-device presentation - 03

Page with Engagement Button



Selection page

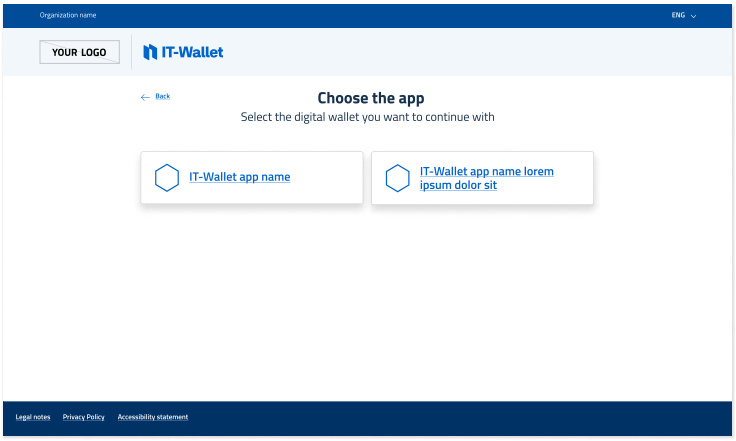
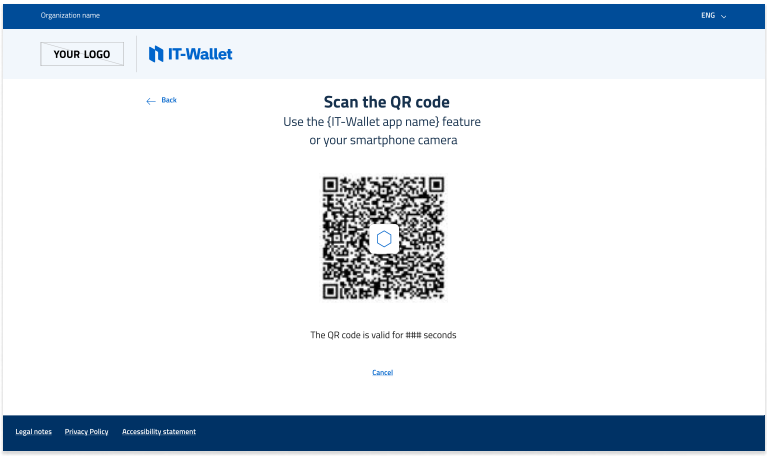


Fig. 5.23: Illustrative page with Engagement Button, Selection page, QR code page, Thank you page for the remote presentation

QR code page



Thank you page

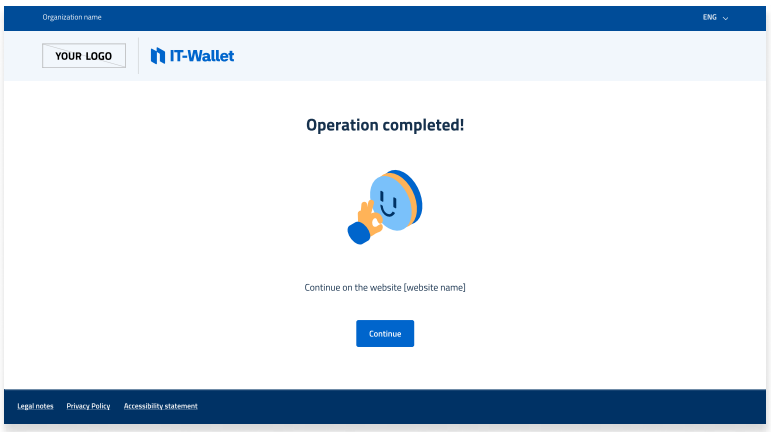


Fig. 5.24: Illustrative page with Engagement Button, Selection page, QR code page, Thank you page for the remote presentation

Both flows are shown below with illustrative wireframes.

Relying Parties SHOULD use the official-resources:Official Resources for design and development. If a Relying Party does not intend to use such open source resources, it MAY independently develop the Technical Solutions enabling the Authentication flow, ensuring that it follows the specifications herein provided.

Note

The images in this section are to be considered illustrative as they are the subject of interface (UI) evolutions.

Relying Parties MUST implement and provide the following pages as part of the Authentication process:

- **Discovery Page**: lists all the available Authentication methods;
- **Selection Page**: shows the User all the Wallet Solutions available in the IT-Wallet System Register and let them choose which one to continue the Authentication process with;
- **QR Code Page** (*cross-device only*): prompts the User to scan a QR code;
- **Thank You Page**: confirms the successful Authentication;
- **Error Page**: displays error messages related to the Authentication process.

Each of these pages MUST include the following recurring elements, in line with the Visual Identity of the Relying Party's Touchpoint:

- A **header and/or subheader** allowing Users to navigate back to the previous page.
- A **footer** including the privacy policy, legal notice, and accessibility statement, where required by current regulations.

Specific requirements for each individual page are detailed below.

Discovery Page

To enable Authentication via the IT-Wallet System, the Relying Party MAY replace its existing Discovery Page with the version provided in the official-resources:Official Resources.

Alternatively, the Relying Party MAY maintain its own Discovery Page but MUST integrate the Authentication Button as specified in the *Authentication Button* (page 90) section.

The Relying Party implementing the page:

- MUST display all available Digital Identity Authentication methods, including the IT-Wallet System Authentication through the Authentication Button;
- MAY also present alternative Authentication methods, if available;

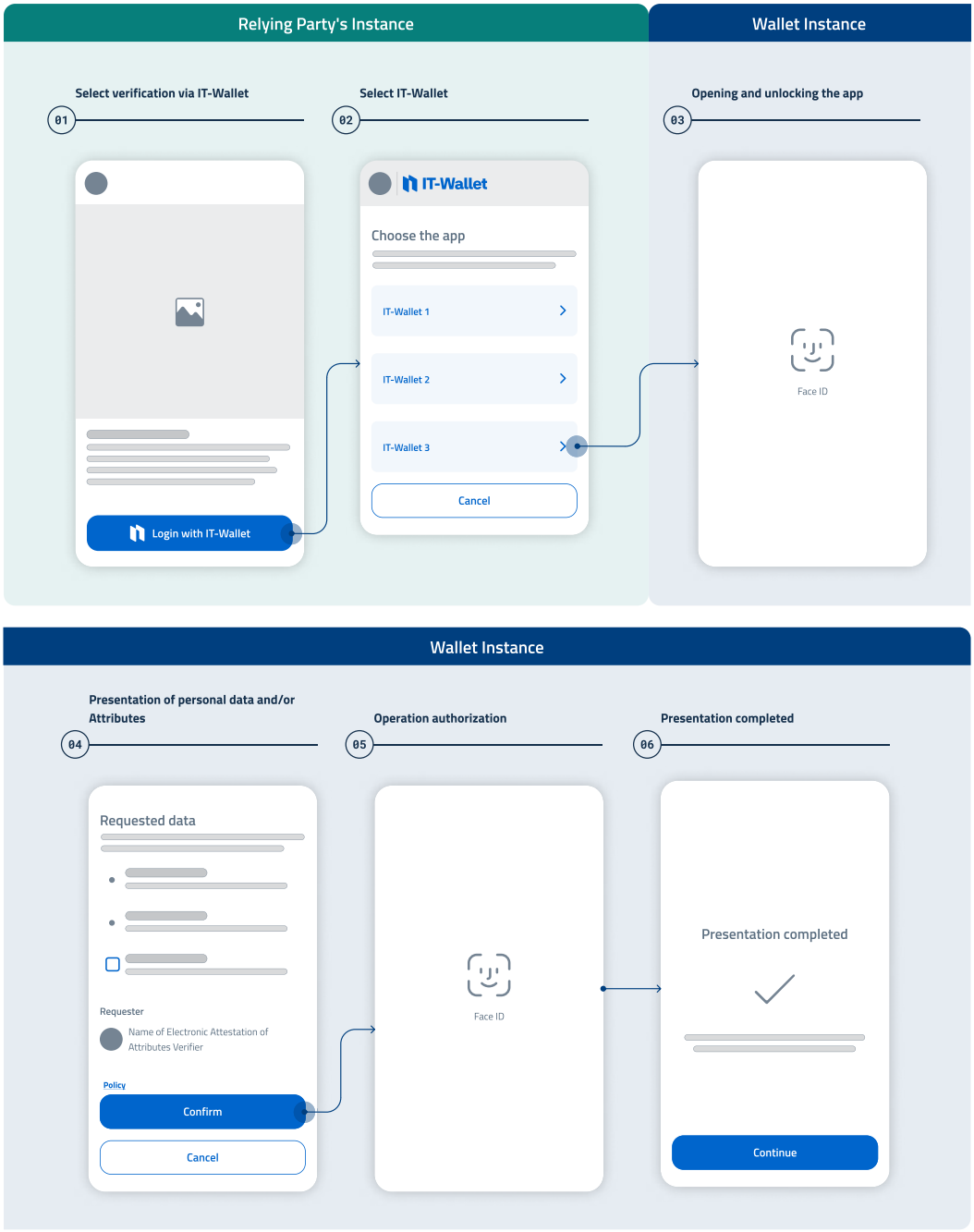


Fig. 5.25: Example of same-device Authentication User Experience - 01

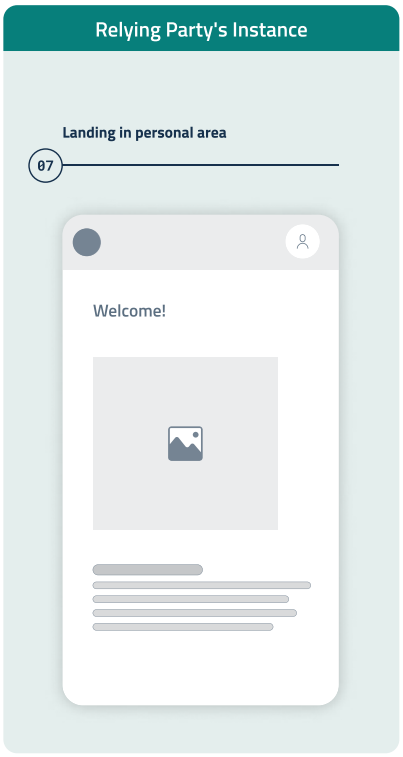


Fig. 5.26: Example of same-device Authentication User Experience - 02

- SHOULD provide essential supporting information to help the User make an informed and conscious choice.

If the User accesses the Discovery Page from a different Touchpoint than the one where the Wallet Instance is activated (cross-device), selecting IT-Wallet System Authentication MUST redirect the User to the QR Code Page.

If the User accesses the Discovery Page from the same Touchpoint where the Wallet Instance is activated (same-device), the selection MUST trigger the opening of the User's Wallet Instance.

Selection Page

The Selection Page is the page on which the User lands after they have chosen to Authenticate via the IT-Wallet System, and is intended to present the User with the Wallet Solutions available to perform Authentication.

Note

This section describes the display of a Selection Page as part of the Authentication process use case. The same page SHOULD be used also by the Relying Parties during presentation and by third parties offering the Credential Offer to enable the Wallet Solution selection

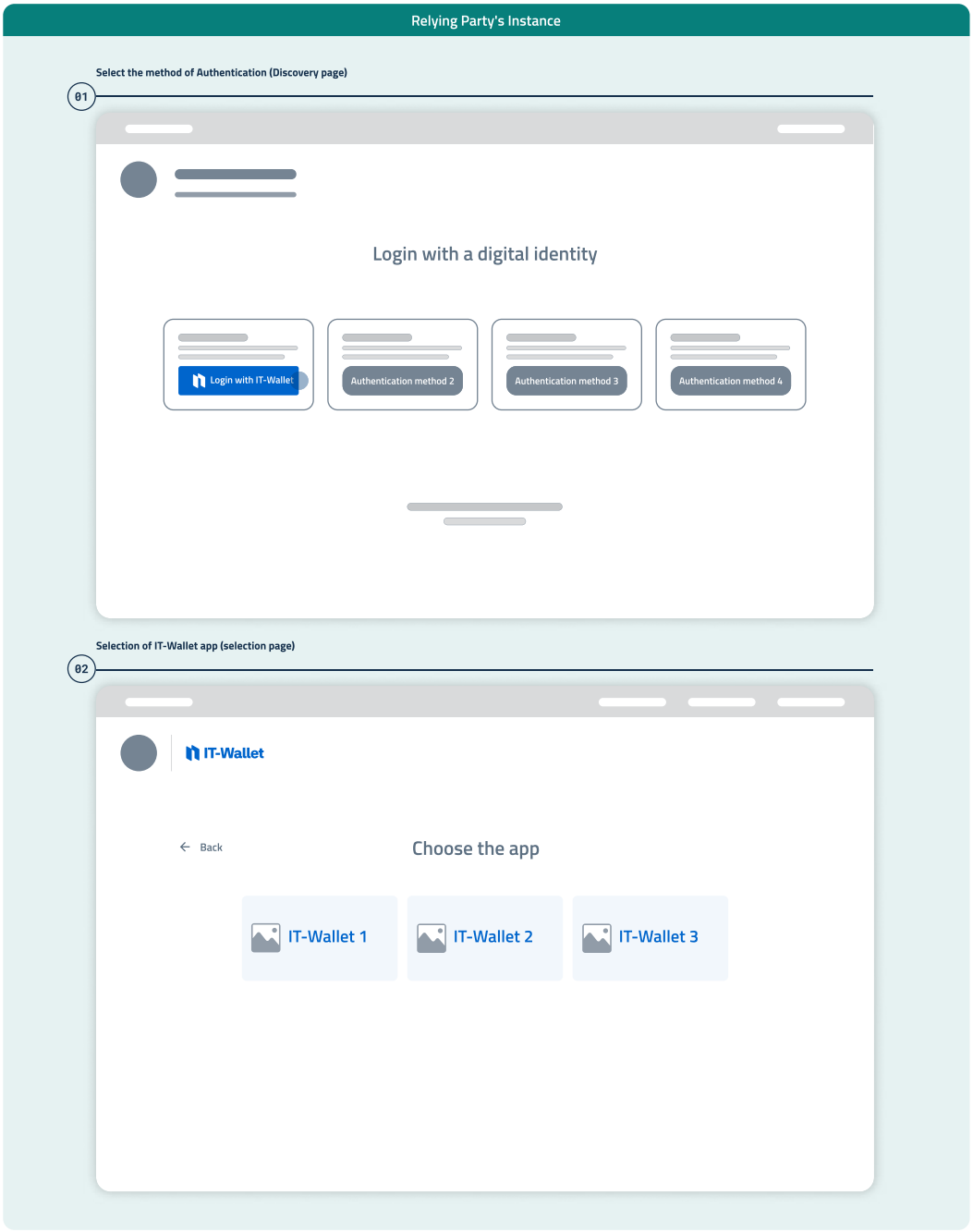


Fig. 5.27: Example of cross-device Authentication User Experience - 01

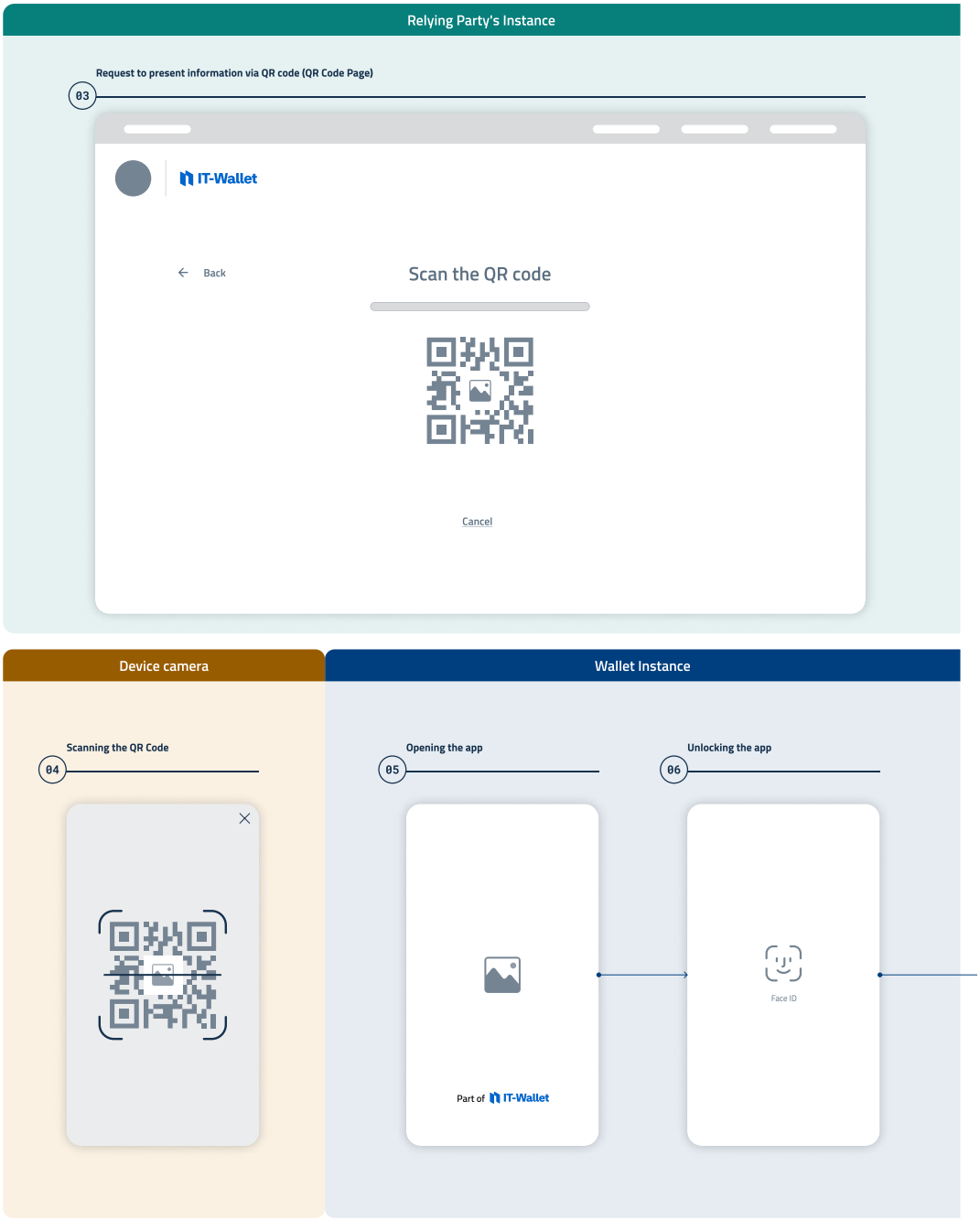


Fig. 5.28: Example of cross-device Authentication User Experience - 02

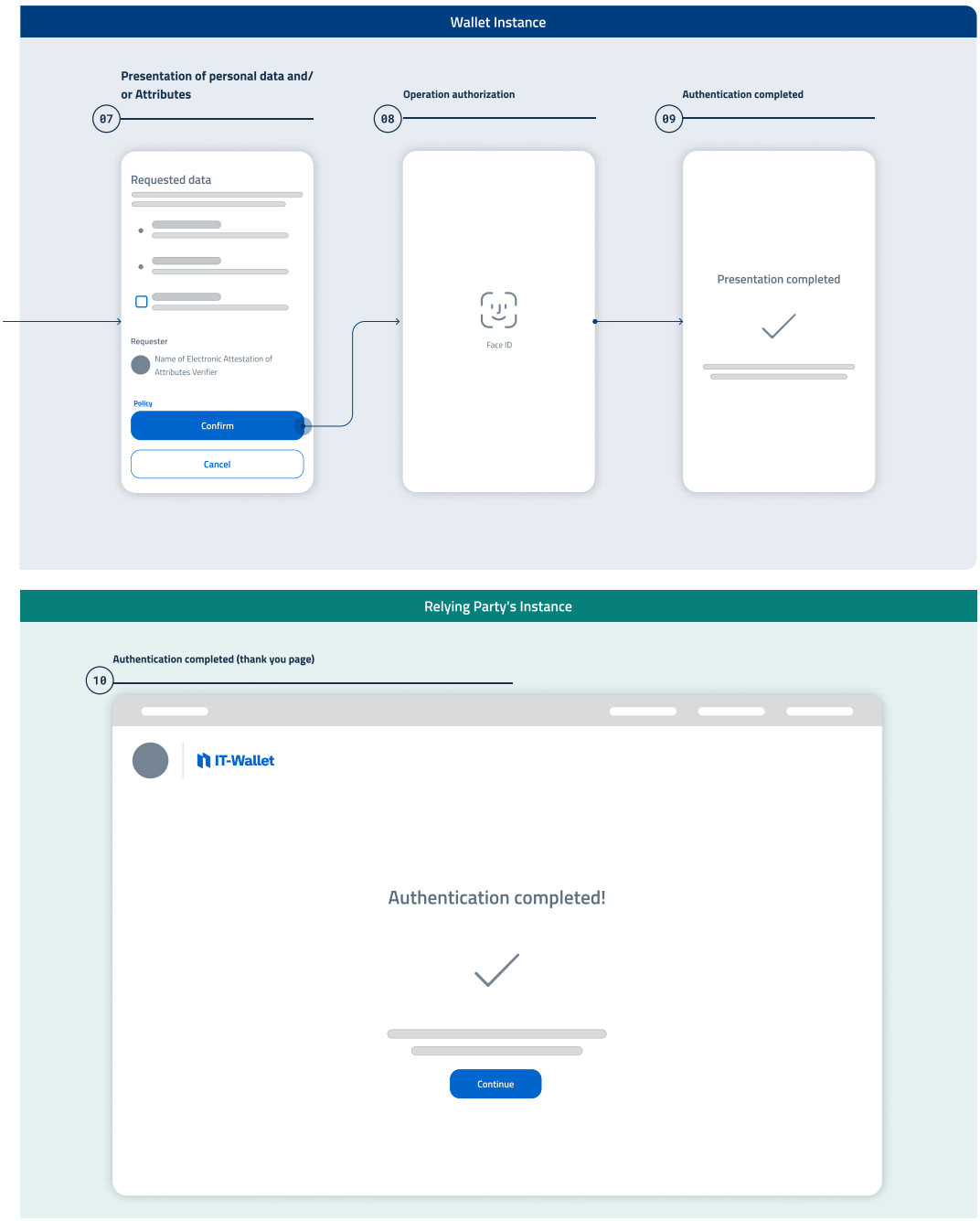


Fig. 5.29: Example of cross-device Authentication User Experience - 03

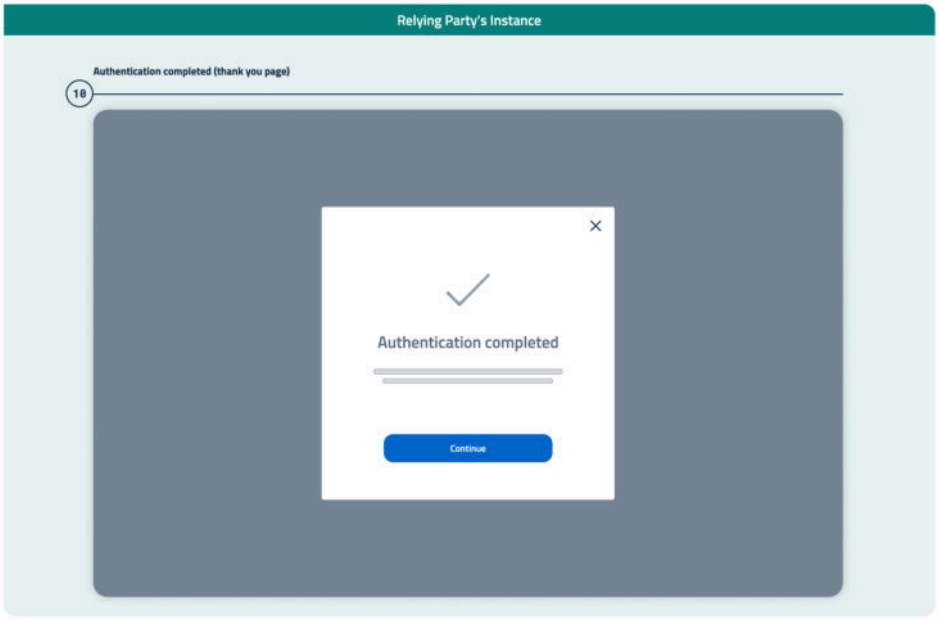


Fig. 5.30: Example of cross-device Authentication User Experience - 04

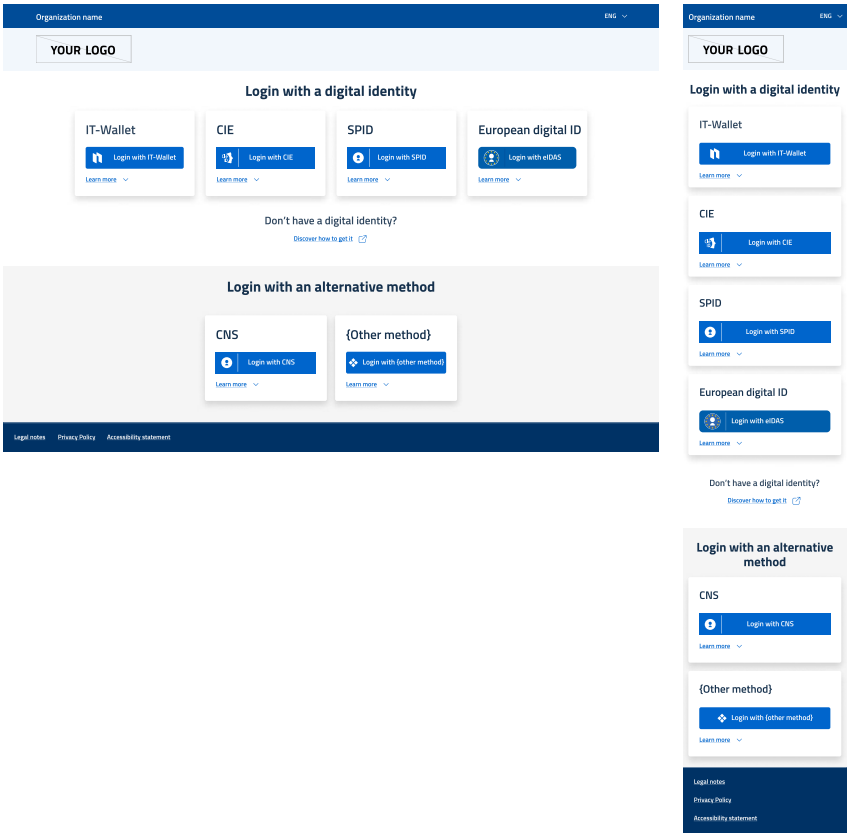


Fig. 5.31: Layout Model of Discovery Page in grid

option. Further details are provided in *Remote Flow* (page 483) and *Credential Offer Flow* (page 452).

The Relying Party MUST implement the Selection Page made available in the official-resources: Official Resources.

The Relying Party implementing the page:

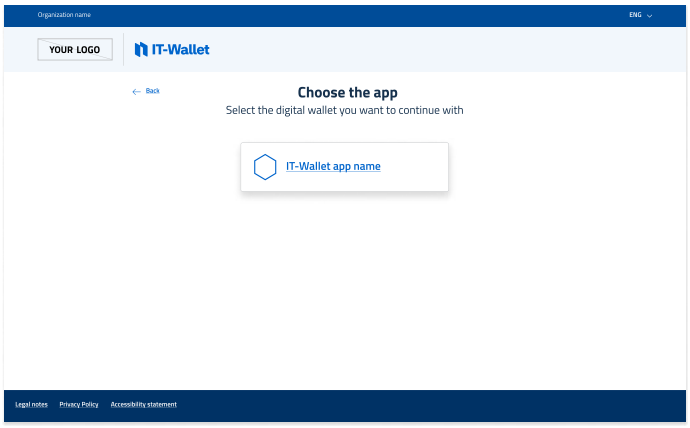
- MUST include the elements proper to the Visual Identity of the IT-Wallet System, including the Logo by placing it alongside its own logo according to the guidance provided in the *Visual Identity* (page 16) section;
- MUST ensure that the copy on the page mirrors that reported in the official-resources: Official Resources;
- MUST present each Wallet Solution in the IT-Wallet System Register through a modular component that displays the logo and name in full retrieved as described in *Wallet Metadata Retrieval Flow* (page 541);
- MUST present the Wallet Solutions in a dynamic layout that adapts to the number of Wallet Solutions available: when the Wallet Solution number is less than 2, the Selection Page MUST distribute the Wallet Solutions within a center-column layout. Otherwise, when the Wallet Solution number is equal or superior than 3, the Selection Page MUST distribute the Wallet Solutions in a 2-column grid; in all cases random sorting MUST be guaranteed;
- MUST allow the User to search for a Wallet Solution through a filter feature by name, when more than 5 Wallet Solutions are present;
- MUST allow the User to find out, when needed, which Wallet Solutions are available in the IT-Wallet System Register by preparing a cross-reference to the official site of the IT-Wallet System;
- MUST include a Call to Action that allows the User to abort the operation and return to the previous page (e.g., the Discovery Page in case of an Authentication process).

The Relying Party MAY also include a text component on the Selection Page to promote the Authentication mode via IT-Wallet, which links back to the IT-Wallet System's official website, as represented in the official-resources:Official Resources.

QR Code Page (cross-device only)

The QR Code Page is presented to the User who selects IT-Wallet System Authentication within a cross-device process. Its purpose is to prompt the User to scan the generated QR code using their Wallet Instance or the camera of their device.

Selection page | 1 Wallet Solution



Selection page | < 3 Wallet Solutions

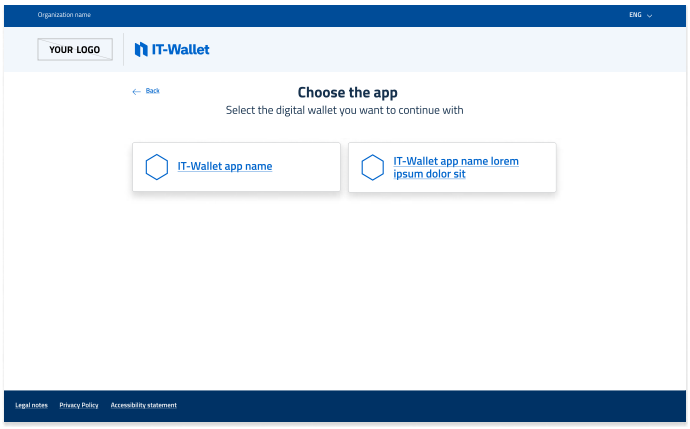


Fig. 5.32: Selection Page desktop

Selection page | + 3 Wallet Solutions

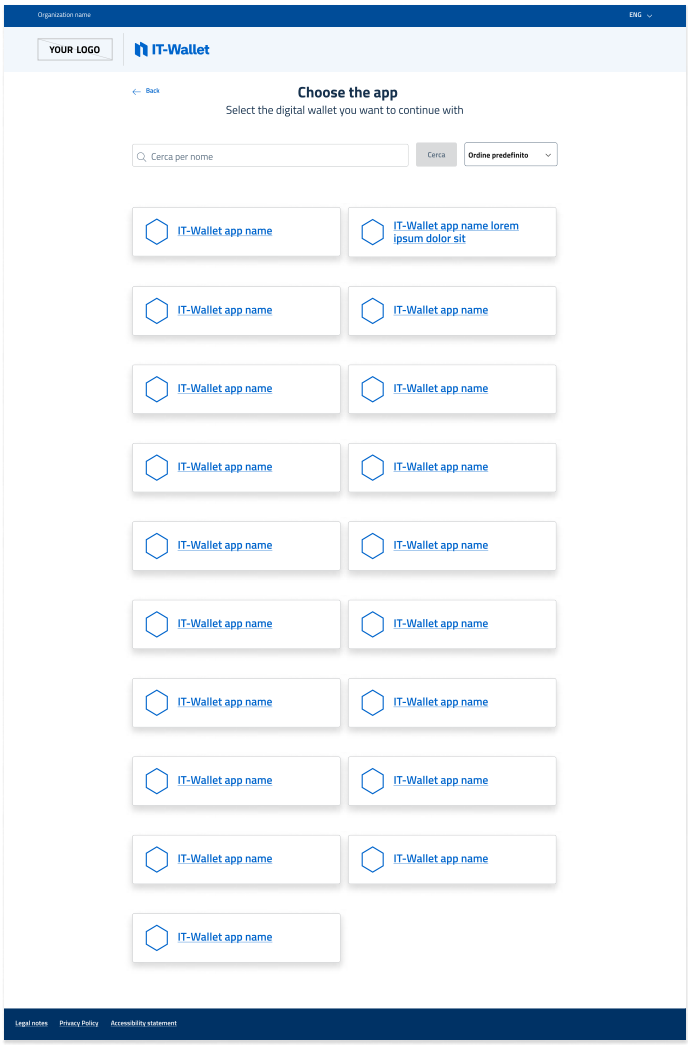


Fig. 5.33: Selection Page desktop

Selection page | Ordering Solutions

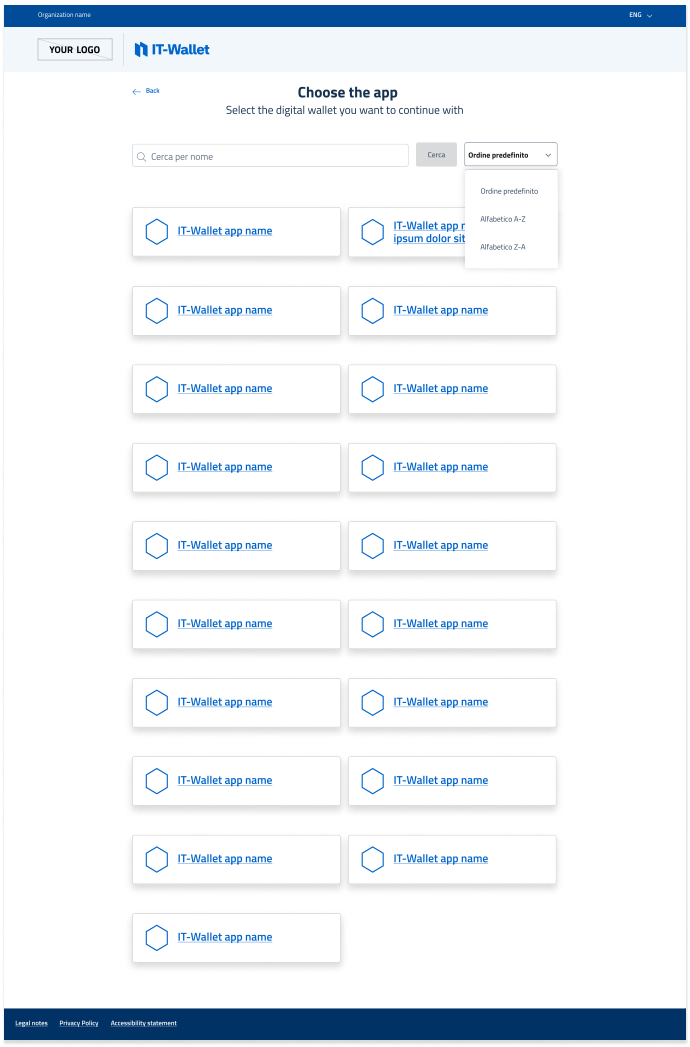


Fig. 5.34: Selection Page desktop

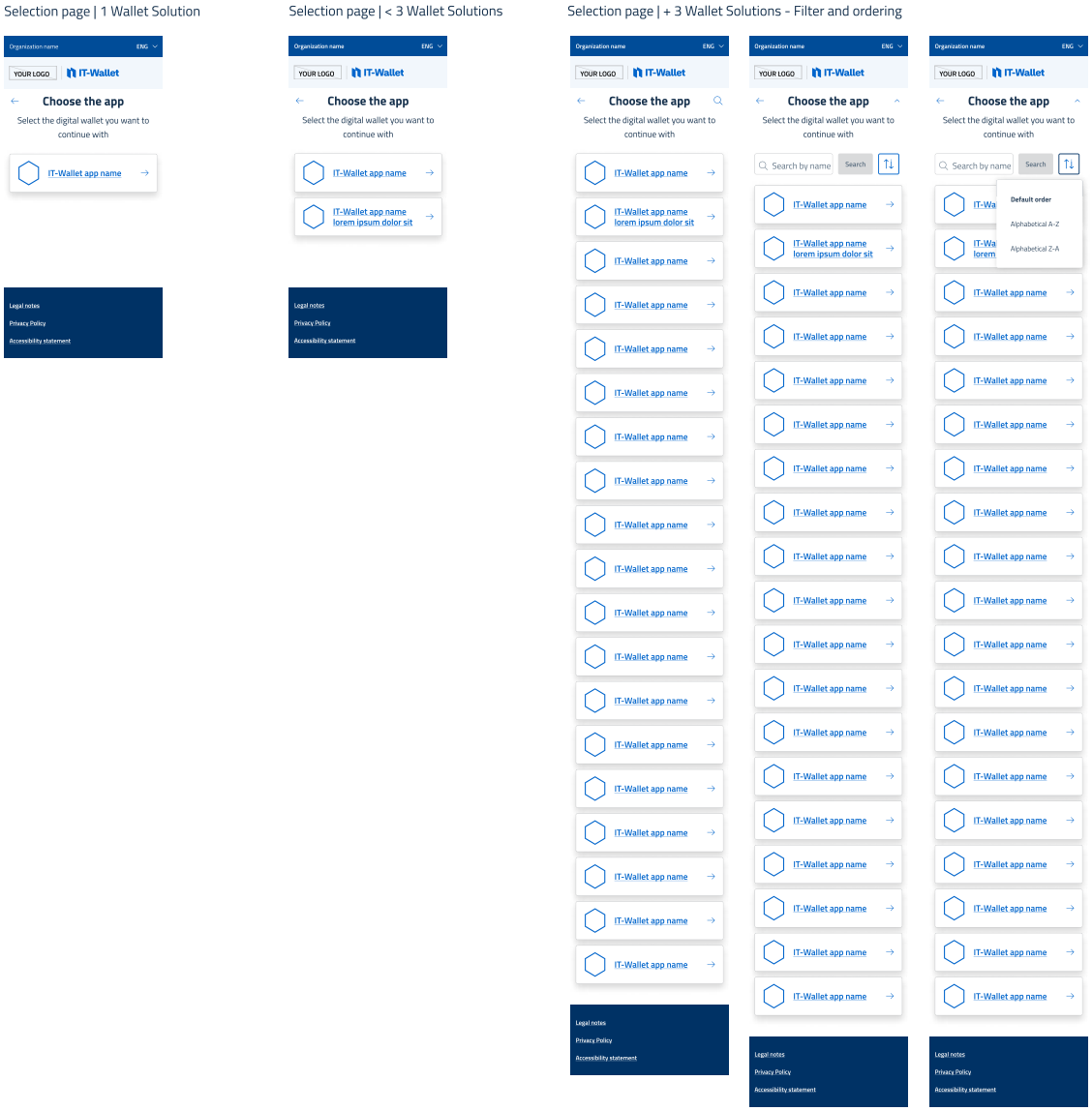


Fig. 5.35: Selection Page mobile

Relying Parties MUST implement the QR Code Page (cross-device) provided in the official-resources:Official Resources.

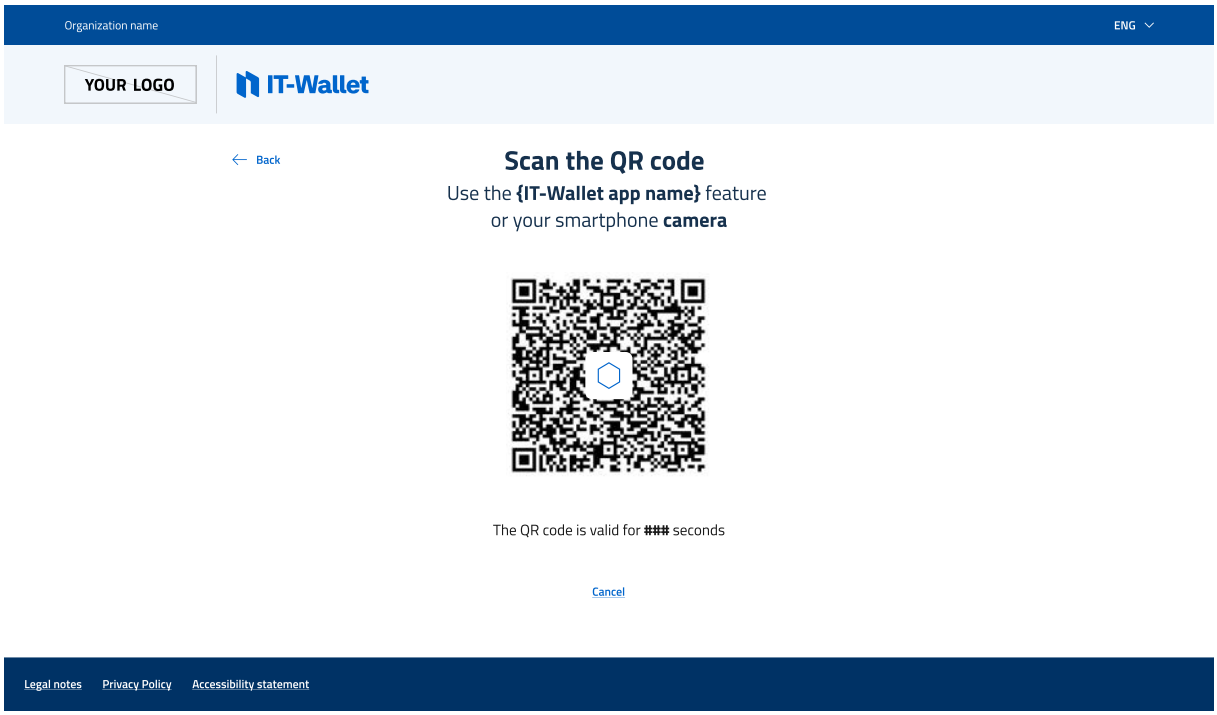


Fig. 5.36: QR Code Page

The Relying Party implementing the page:

- MUST include the Visual Identity elements of the IT-Wallet System, including the logo;
- MUST ensure that the copy on the page mirrors that reported in the official-resources: Official Resources;
- MUST include a Call To Action allowing the User to generate a new QR code, if it has an expiration;
- MUST include a Call To Action allowing the User to cancel the operation and return to the Discovery Page.

Furthermore, in compliance with [REF_ACCESSIBILITY], regarding the QR code, the Relying Parties:

- MUST respect the minimum recommended dimensions to ensure effective scanning. A size of 150x150 pixels is generally adequate, but for codes with high data density (e.g. long URLs or numerous characters), it is advisable to increase it to 300x300 pixels or more;
- MUST ensure minimum contrast between the QR code and the background (the ideal condition provides for a white background with a black QR code);
- MUST avoid color inversions between background and QR code;

- MUST limit the presence to only one QR code per page;
- MUST ensure sharpness and high quality;
- MUST ensure SVG format;
- MUST ensure that it is not partially hidden by text or other elements.

Thank You Page

The Thank You Page is displayed after the User completes the Authentication process via their Wallet Instance. Its purpose is to inform the User about the successful Authentication.

Relying Parties MUST implement the Thank You Page provided in the official-resources:Official Resources.

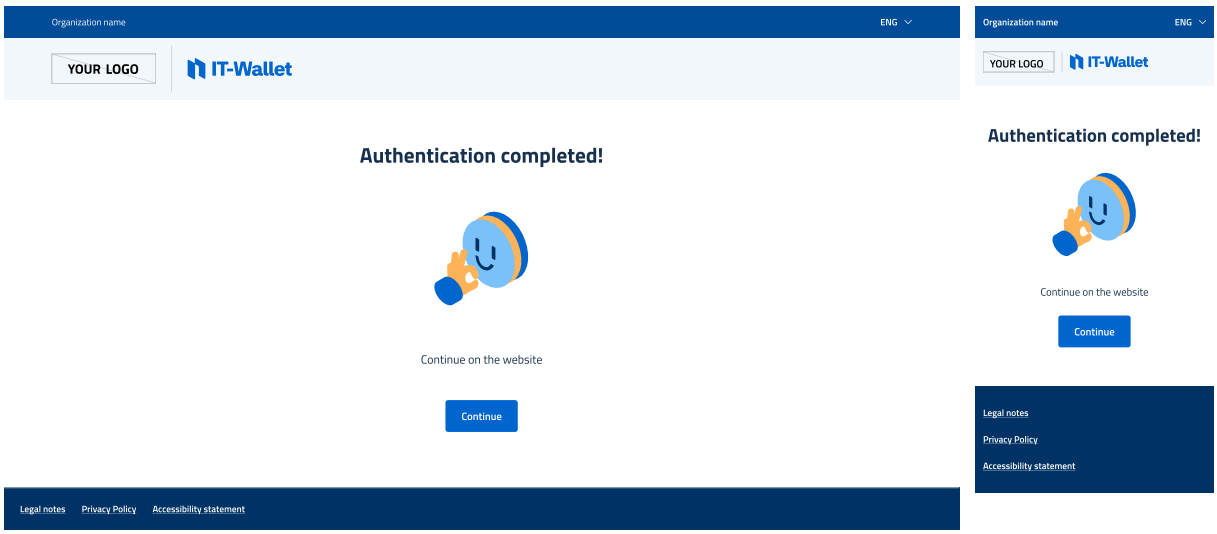


Fig. 5.37: Thank You Page

The Relying Party implementing the page:

- MUST include the Visual Identity elements of the IT-Wallet System, including the logo and an icon or graphical element consolidating the message;
- MUST ensure that the copy on the page mirrors that reported in the official-resources: Official Resources;
- MUST include a Call To Action prompting the User to proceed to the Touchpoint authenticated area.

Error Page

The Error Page is displayed when an issue occurs during the Authentication process. Its purpose is to inform the User about the nature of the error (e.g., technical issue, network issues, Wallet Instance malfunction, denied data sharing, etc.) and to present the available next steps. For further details, refer to the *Error Management* (page 103) section.

Relying Parties MUST implement the Error Page provided in the official-resources:Official Resources.

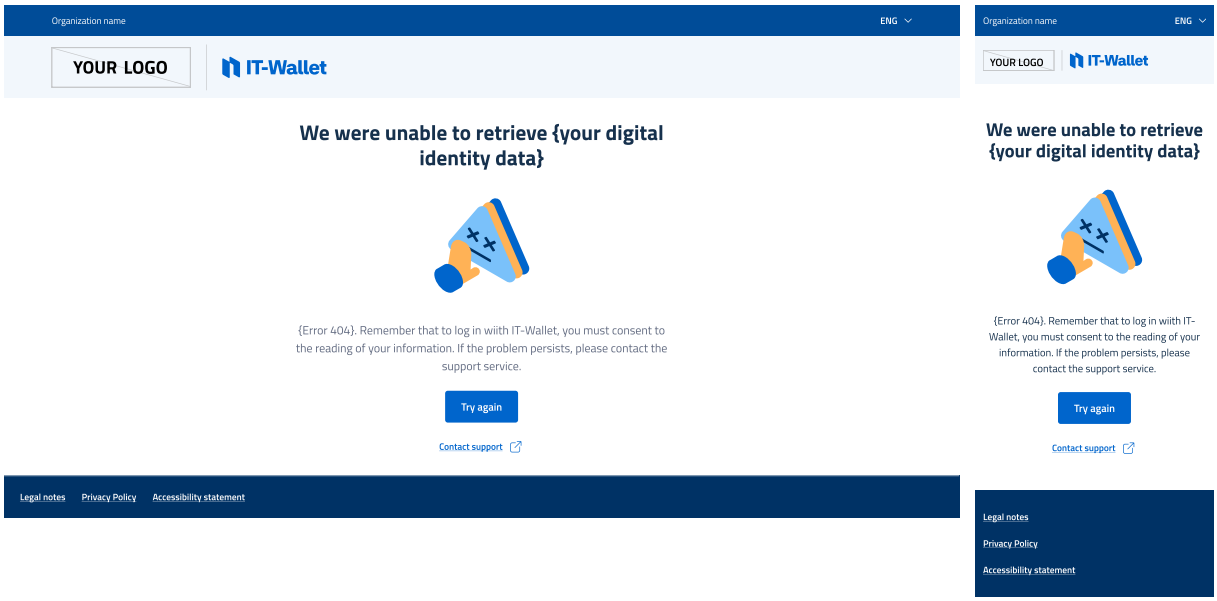


Fig. 5.38: Error Page

The Relying Party implementing the page:

- MUST include the Visual Identity elements of the IT-Wallet System, including the logo and an icon or graphical element that conveys the type of error;
- MUST ensure that the copy on the page mirrors that reported in the official-resources: Official Resources;
- MUST include one or more Call To Action guiding the User toward the appropriate next step (e.g., retry, contact support, etc.).

Authentication Button

The Authentication Button "Login with IT-Wallet" serves as an *Engagement Button* (page 49), providing Users with a standardized way to Authenticate themselves using their digital Wallet.

Relying Parties MUST make the Authentication Button available within the Discovery Page of their Technical Solutions to allow the User to get authenticated into their services using the Wallet Instance under their control.

The Authentication Button has the following requirements:

- The Authentication Button MUST be used exactly as outlined in the official-resources: Official Resources and MUST NOT be redesigned ad hoc;

- The Authentication Button MUST be used only in the shapes, colors and proportions defined and MUST NOT be altered, distorted, or hidden;
- The Authentication Button MUST be responsive to all screen resolutions and MUST be integrated in the Discovery Page in order to meet minimum usability and accessibility requirements;
- Actors wishing to integrate the Authentication Button into their Technical Solution MUST ensure that it is translated into other languages, at least English;
- The Authentication Button MUST maintain a minimum distance from other elements (`quiet zone`) of at least 24px;
- The Authentication Button MUST state "Login with IT-Wallet";
- The Authentication Button SHOULD always be accompanied by an external link (e.g., "Learn more") that links to the official website of the IT-Wallet System, indicated in the official-resources:Official Resources section;
- Where space allows and/or the context requires it, the Authentication Button SHOULD be accompanied by a descriptive text: "IT-Wallet is the Italian digital wallet system that allows you to authenticate online and access public and private services quickly and securely, using the data, documents, and attestations you can obtain digitally in one of the participating wallets.".

Below are some non-mandatory examples of Authentication Button layout:

The integration of the Authentication Button within the Discovery Page may vary depending on the page layout. Below are illustrative, non-exhaustive examples of Discovery Pages using grid, tab, and list layouts, respectively.

For further details on the use of the Authentication Button, please refer to the *Authentication* (page 69) section.

"Login with IT-Wallet" button - html code

The button is available in three variants of dimensions (S - default / M / L) as per the .Italia Design System, and in "get" (call to an external page) and "post" (form inside the button) formats. In addition to the variants of dimensions, are provided two variants of button with fixed width, to be used in situations where it's preferable to keep consistency between similar buttons:

- the fixed-width version with icon on the left and centered text;
- the fixed-width version with icon and text centered.

The Official Resource of the Authentication Button is available in the official-resources:Official Resources of these Technical Specification.

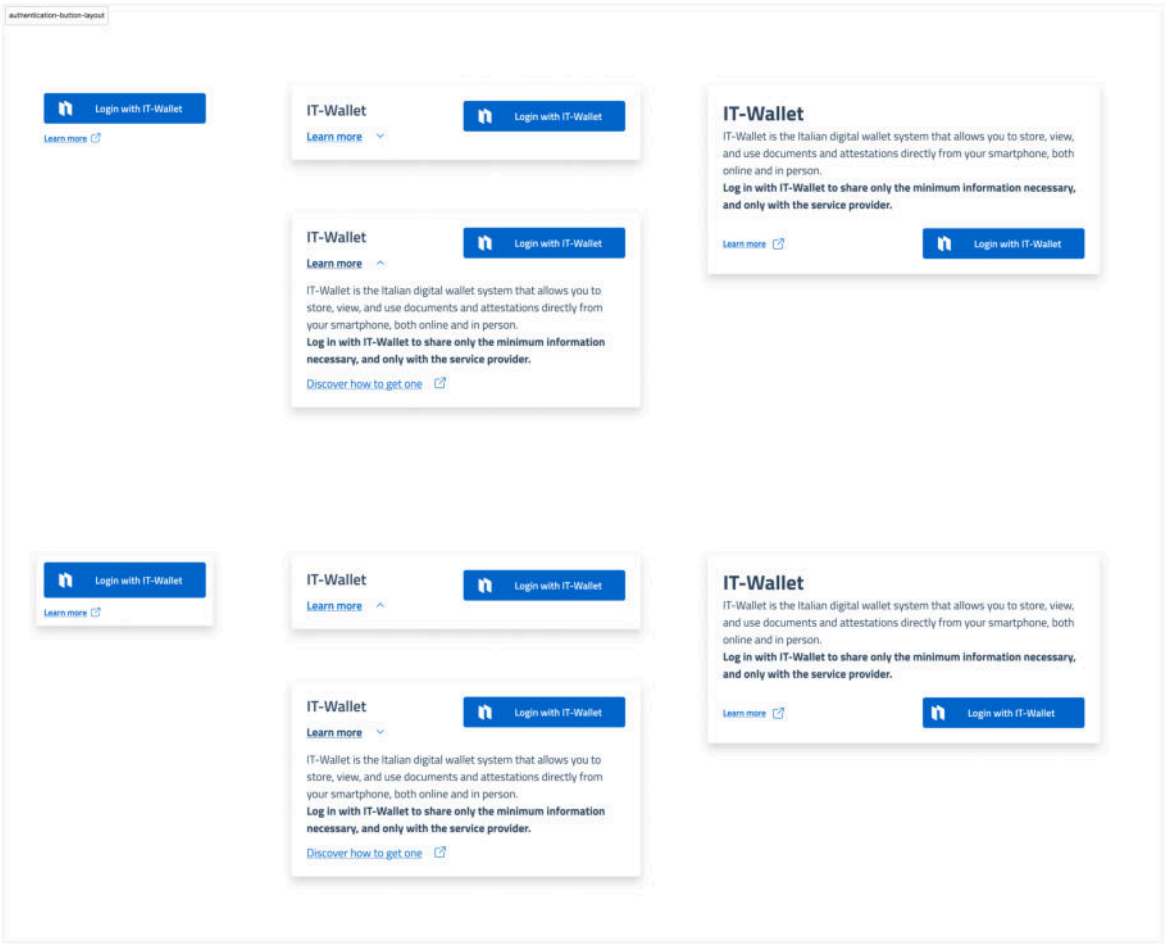


Fig. 5.39: Variants of Authentication Button layout

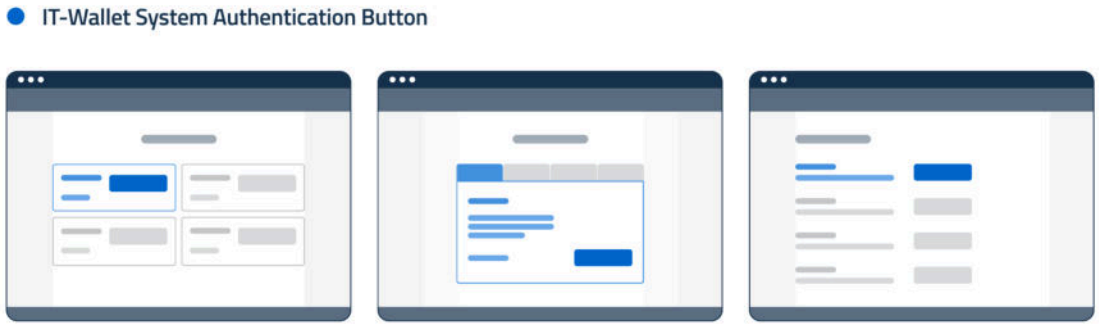


Fig. 5.40: Examples of Discovery Page layouts: grid, tab, and list

"Login with IT-Wallet" button – svg For further information on the Authentication button refer to the Brand Manual, indicated in the official-resources:Official Resources section. The Authentication button Official Resource is available in the official-resources:Official Resources section of these Technical Specifications.

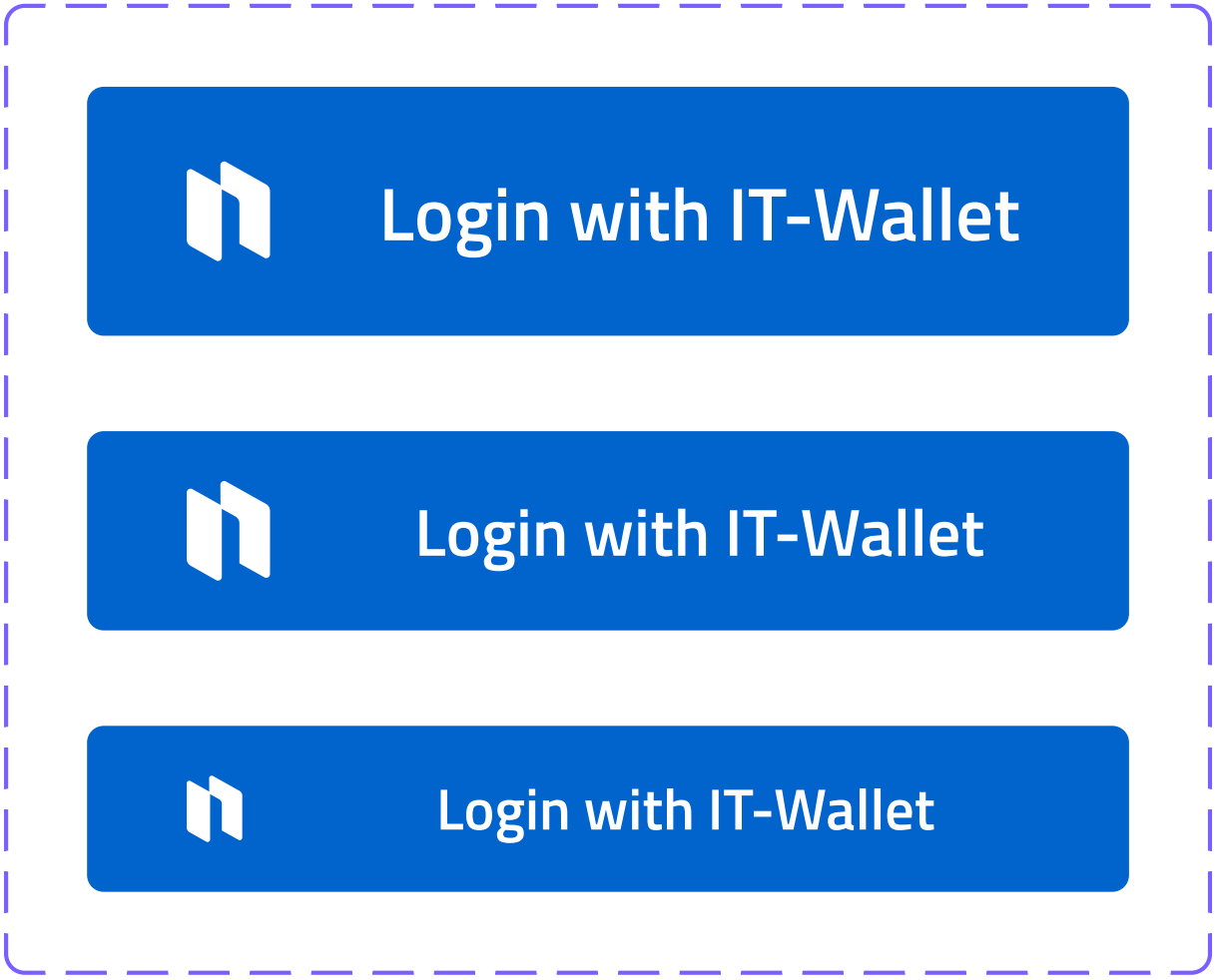


Fig. 5.41: Example of Authentication button in its dimension variants



Fig. 5.42: Authentication button fixed justified



Fig. 5.43: Authentication button fixed centered

5.6 Management of Electronic Attestations

The Wallet Provider, via their Wallet Solution, and the PID provider or Electronic Attestations of Attributes Provider, via dedicated Touchpoints, MUST let the User manage their Electronic Attestations at any time.

This section outlines three different categories of requirements for managing each Electronic Attestations, specifically regarding:

- **Its status**: to allow the User to verify whether an Electronic Attestation is valid or invalid;
- **Its usage**: to enable the User to view and manage the history of presentations carried out with an Electronic Attestation;
- **Its data**: to allow the User to backup and restore each Electronic Attestation of Attributes in compliance with the principle of data portability.

Below are the key aspects that impact and define the User Experience in managing Electronic Attestations though the Wallet Instance, along with the functional requirements associated with each category.

5.6.1 Status of Electronic Attestations

To ensure reliability and promote the proper use of a Wallet Solution, the Wallet Provider MUST guarantee the User to always have visibility of the status of the Electronic Attestations issued within their Wallet Instance, based on the information received from the Electronic Attestations Provider, which manages their lifecycle.

The admissible statuses for an Electronic Attestation are as follows:

- **Valid** (`valid`): the EAA is issued without any evidence of criticalities or issues. In this case, the Wallet Instance MUST guarantee the User full presentation functionalities, both in proximity and remote scenarios;
- **Suspended** (`suspended`): the EAA is temporarily invalid, in a reversible condition (e.g., suspended driving license). In this case, the Wallet Instance:
  - MUST provide the User with adequate evidence of this status;

  - MUST invite the User to wait for the EAA to automatically return to a valid status at the end of the suspension period;
  - MUST guarantee the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.

- **To be Updated** (`update` or `attribute_update`): the EAA is invalid because one or more of its information elements, at the metadata or Attribute level, is obsolete. In this case, the Wallet Instance:
  - If one or more of the EAA metadata is obsolete:
    * SHOULD automatically receive the EAA update managed by the Electronic Attestations Provider without informing or requiring action from the User or cause service interruptions in terms of EAA management and presentation.
  - If one or more of the EAA Attributes are obsolete:
    * MUST provide the User with adequate evidence;
    * MUST invite the User to update the EAA through a new issuance process;
    * MUST guarantee the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.

- **Invalid** (`invalid`): the EAA is invalid, in an irreversible condition (e.g., revoked driving license). In this case, the Wallet Instance:
  - MUST provide the User with adequate evidence;
  - MUST invite the User to update or delete the EAA depending on whether they are entitled to re-obtain it and, therefore, use it again (e.g., a renewed driving license – which can be re-obtained as an EAA – or an exhausted medical prescription – which cannot be re-obtained as an EAA);
  - MUST guarantee the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.

In addition to the statuses listed above, it should be specified that an EAA is subject to expiration. Two types of expiration are distinguished:

- **Administrative expiration**: characterizes certain EAAs and is provided by the Authentic Source within the Attributes of the EAA itself (e.g., driving license expiration date);
- **Technical expiration**: characterizes all EAAs and is defined by the Authentic Source in synergy with the Electronic Attestations Provider with the aim of mitigating security

risks. This expiration is generally set at 1 year or, in any case, at a period less than or equal to the administrative expiration date.

Consequently, the EAA assumes the following additional statuses:

- **Expiring**: the EAA is valid but close to the administrative expiration date, if made available by the Authentic Source, or to the technical expiration. In this case, the Wallet Instance:
  - If the administrative expiration is approaching (e.g. 30 days remaining):
    * SHOULD provide the User with adequate evidence;
    * SHOULD invite the User to perform any necessary actions to re-obtain the updated EAA (e.g., renew the driving license at the competent offices);
    * MUST guarantee the User presentation functionality, both in proximity and remote scenarios.
  - If the technical expiration is approaching (e.g. 7 days remaining):
    * SHOULD automatically trigger the EAA update managed by the Electronic Attestations Provider without requiring action from the User.
    * If the automatic update is successful and returns an updated EAA:
    * MUST replace the previous one without resulting in unexpected duplicates, disruption or notification to the User;
    * If the automatic update is successful but does not return any EAA:
    * MUST provide the User with adequate evidence of the potential loss of entitlement to the EAA itself;
    * MUST invite the User to update or delete the EAA depending on whether they are entitled to re-obtain it and, therefore, use it again (e.g., a renewed driving license – which can be re-obtained as an EAA – or an exhausted medical prescription – which cannot be re-obtained as an EAA);
    * MUST guarantee the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.
    * If the automatic update fails due to the service being unavailable or other technical errors:
    * SHOULD attempt the update at least once more before the EAA reaches its technical expiration date.

- **Expired**: the EAA is valid but has passed the administrative expiration date, if provided by the Authentic Source, or has passed the technical expiration date. In this case, the Wallet Instance:
  - If the administrative expiration date has passed:
    - MUST provide the User with adequate evidence;
    - MUST invite the User to perform any necessary actions to re-obtain the updated EAA (e.g., renew the license at the competent offices);
    - MUST guarantee the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.
  - If the technical expiration date has passed (automatic update not performed or failed):
    - MUST provide the User with adequate evidence, distinguishing this specific case from an administrative expiration;
    - MUST invite the User to update the EAA by initiating an EAA re-issuance flow;
    - MUST allow the User presentation functionality, both in proximity and remote scenarios. Consequently, the Relying Party Instance MUST adequately inform the Verifier about the specific status of the EAA.

Below are the functional requirements supporting the User Experience regarding the update of the Electronic Attestation that the Electronic Attestation Provider MUST guarantee through the Wallet Solution:

- The User sees in the Electronic Attestation Preview View that its status is not valid;
- The User sees a message in the Detail View informing them of the new status of the Electronic Attestation and MAY find out more information;
- The User sees any additional information on the requirements and/or limitations relating to the status of the Electronic Attestation of Attributes and MAY close the message or proceed with any action requested by the Electronic Attestation Provider;
- The User sees specific Call To Action buttons in the Detail View to delete the Electronic Attestation or update it if no longer valid.

The flow is shown below with illustrative wireframes.

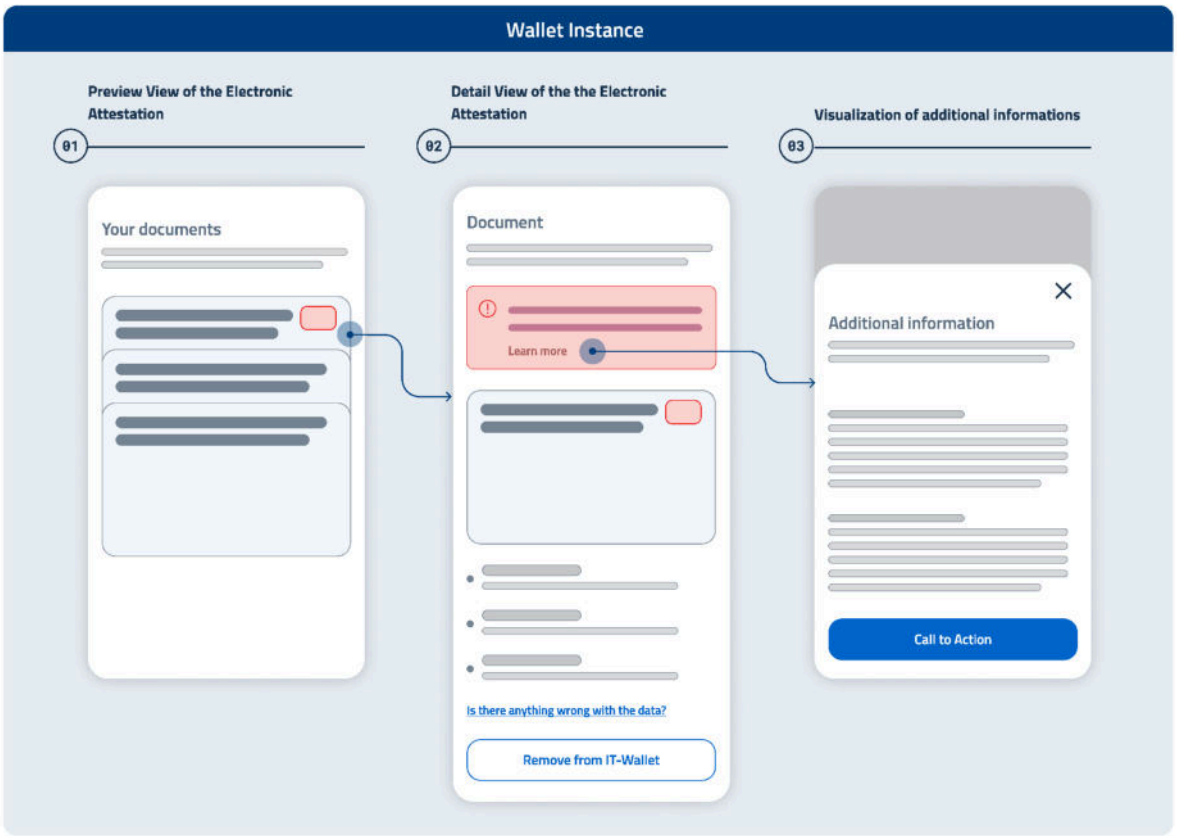


Fig. 5.44: Example of User Experience in Updating an Electronic Attestation

Revocation of Electronic Attestations

Revocation is the procedure that turns an Electronic Attestation from a valid state to an invalid state. Revocation can occur in either an active or passive mode:

- **Active revocation**: This refers to the revocation of an Electronic Attestation at the User's request. This process affects only the Electronic Attestation and not its corresponding physical document, if one exists. Below is an illustrative list of scenarios in which the Wallet Provider MUST give the User the ability to request the revocation of an Electronic Attestation:
  - The User decides they no longer wish to use a specific Electronic Attestation;
  - The User decides to deactivate their Wallet Instance, thereby revoking all previously obtained Electronic Attestations;
  - The User no longer has possession of the device on which their Wallet Instance is installed due to loss or theft.
- **Passive revocation**: This refers to the revocation of an Electronic Attestation managed by the respective Electronic Attestation Provider on behalf of the Authentic Source. In this case, the Wallet Instance MUST inform the User of the status change of the Electronic Attestation and the Electronic Attestation Provider MAY additionally notify the User via other Touchpoints . Below is an illustrative list of scenarios that could lead to the revocation of an Electronic Attestation:
  - The physical document corresponding to the Electronic Attestation has been reported lost or damaged by the User through the appropriate channel/ Touchpoint;
  - The physical document corresponding to the Electronic Attestation has been revoked by the competent authorities;
  - The minimum security and/ or reliability requirements for one or more involved parties are no longer met.
  - The User's device no longer meets the minimum security requirements (rooted or jailbroken).

Below are the User Experience requirements that the Wallet Provider MUST guarantee via their Wallet Solution: - The User opens the Detailed view of the Electronic Attestation that they want to revoke; - The User selects the Call to Action to revoke the Electronic Attestation; - The User reviews all information relevant to the action being performed and either gives their consent to proceed or denies consent to cancel the operation; - The User sees the successful outcome of the revocation.

The flow is shown below with illustrative wireframes.

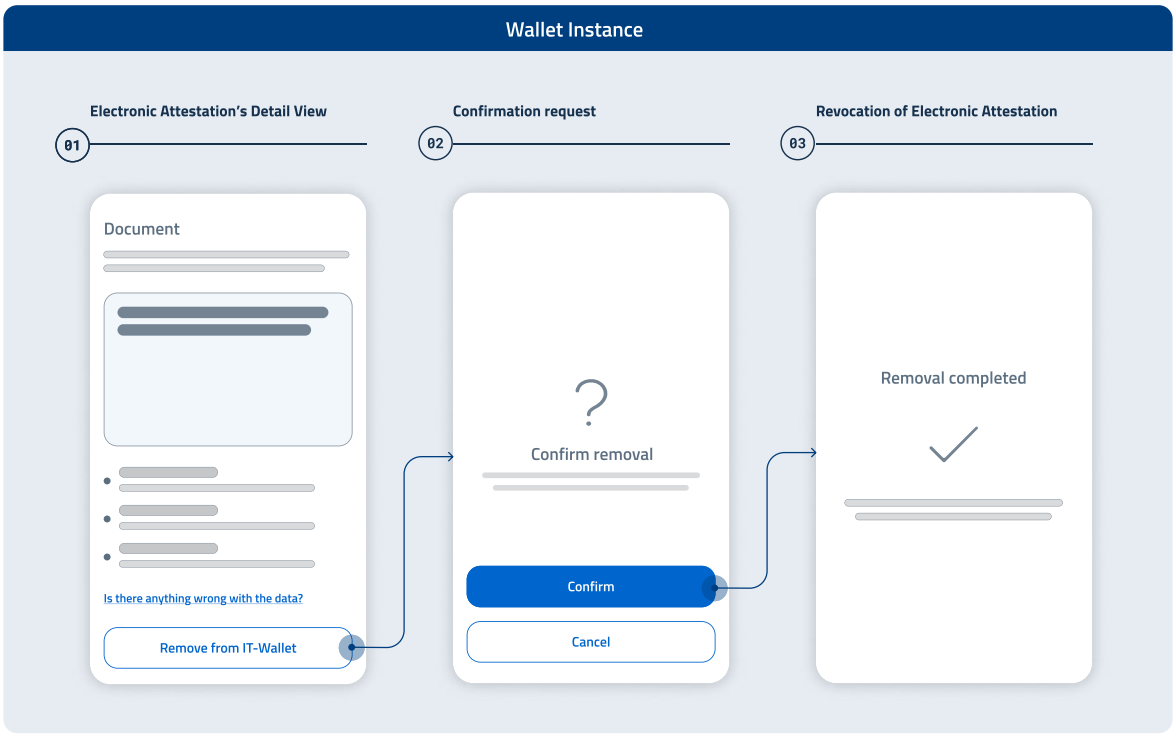


Fig. 5.45: Example of User Experience in Revoking an Electronic Attestation.

5.6.2 Transaction Logging

To ensure the principles of visibility and transparency, the Wallet Provider MUST provide a user-friendly dashboard in the Wallet Instance that allows the User to view the history of transactions performed using their Wallet Instance (e.g., Electronic Attestation issuance or presentation). In particular, the dashboard MUST:

- provide an overview of all recorded transactions and allow the User to access detailed views of individual transactions;
- enable the User, where a transaction involves a Relying Party, to easily initiate a data deletion request to the corresponding Relying Party (using logged contact information);
- support export of one or more transaction records to a file;
- allow the User to delete one or more transaction records with prior warning.

5.6.3 Backup and Restore of Electronic Attestation of Attributes

With the aim of ensuring the principle of data portability, the Wallet Solution MUST guarantee the User to have access to specific functionalities, particularly to:

- Request the backup and storage of Electronic Attestations of Attributes obtained through their Wallet Instance;
- Request the restore of their Electronic Attestations of Attributes on another Wallet Instance.

5.7 Deactivation of the Wallet Instance

The deactivation of the Wallet Instance is the functionality that makes the Wallet Instance inactive and therefore no longer operational. The deactivation process can be triggered by different actors depending on the circumstances, specifically:

- By the User, in cases such as:
  - The device has been lost or stolen;
  - The device has been compromised;
  - The device has been reset to factory settings.
- By an authorized third party, in cases such as:
  - The Wallet Solution no longer meets the minimum security requirements.

The Wallet Provider MUST guarantee the User the ability to voluntarily deactivate their Wallet Instance through:

- The Wallet Instance itself;
- A Touchpoint (e.g., a website) provided by the Wallet Provider;
- The device's app store, by uninstalling the Wallet Instance.

Below are the functional and User Experience requirements that the Wallet Provider MUST guarantee via their Wallet Solution:

- The User accesses their Wallet Instance using the previously configured unlock method or Authenticates at the Touchpoint provided by the Wallet Provider;
- The User selects the Wallet Instance deactivation functionality;
- The User is informed that deactivating the Wallet Instance will invalidate previously obtained Electronic Attestations;
- The User confirms the action to proceed with deactivation, or cancels the operation;

- The User receives confirmation of successful deactivation;
- The User is notified that the Wallet Instance is inactive when logging in again.

The User has the ability to reactivate the Wallet Instance by re-downloading the app from the app store (if uninstalled) and/or by following the activation process again. For further details, please refer to the *Activation of the Wallet Instance* (page 37) section.

Once the Wallet Instance is reactivated, Electronic Attestations of Attributes can be re-obtained by starting the issuance or restore process again. For more details, please refer to sections *Issuance of Electronic Attestations of Attributes* (page 42) and *Backup and Restore of Electronic Attestation of Attributes* (page 101).

In case of errors using the Wallet Instance, the Wallet Provider MUST guarantee that the User receives consistent messages that inform them and guide them toward resolving the issue. For more details, please refer to the *Error Management* (page 103) section.

The flow is shown below with illustrative wireframes.

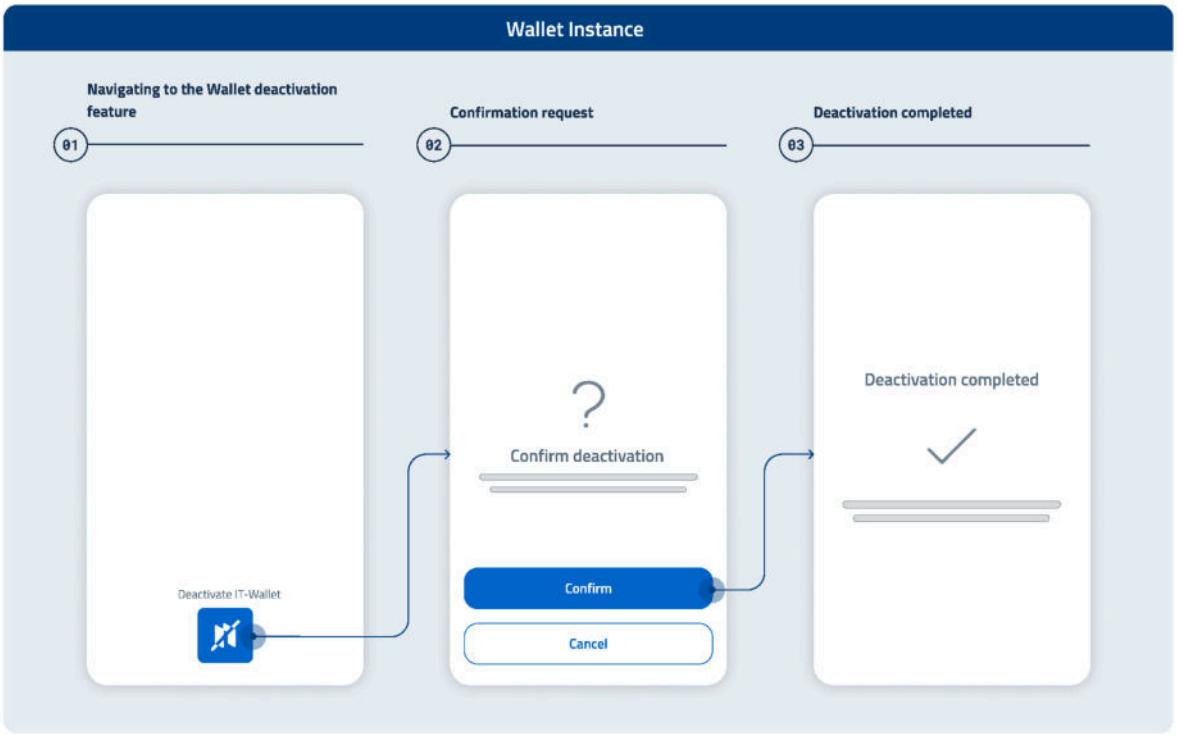


Fig. 5.46: Example of User Experience in Deactivating a Wallet Instance

5.8 Error Management

The IT-Wallet System involves the interaction of multiple services provided by different actors. It is therefore important to define an effective error management model with the goal of improving the perception and reliability of the entire ecosystem and enabling the User to feel guided during interactions with the various Technical Solutions and to consciously manage any issues while using the service.

Effective communication in case of an error also provides benefits for the actors involved, as it contributes to the reduction of assistance requests and, thus, to the minimisation of the impact on support systems.

Each Primary Actor MUST implement a proper error management, in compliance with current Technical Specification, in order to communicate errors and exceptions to the User and through the IT-Wallet Instance. Errors can be categorized, based on their nature, as follows:

- **The stage of the User Experience** where the error may occur: activation or deactivation of the Wallet Instance, obtaining, presenting, or managing Electronic Attestations of Attributes;
- **The type of error**: system error, communication error between actors, etc.;
- **The actor responsible** for the error: Wallet Provider, PID Provider, Electronic Attestations of Attributes Provider, Authentic Source;
- **The way the error is displayed**: message on the page, banner, toast message, and so on;
- **Suggested actions for the User** to resolve the error: suggestion to wait, request to try again, referral to FAQs and/or customer care, etc.;
- **The method for error management**: opening an assistance request through the Wallet Instance, linking to other detailed channels, and so on. For further details, please refer to the *User Assistance* (page 105) section.

Below is a non-exhaustive list of the main error cases, with reference to the actor responsible for their management, for each phase of the User Experience. The detailed list of errors to be managed for each interaction endpoint with the User is available in the dedicated error sections within *Endpoints* (page 543).

5.8.1 Activation of the Wallet Instance Errors

Error type	Actor in charge
The device does not support the Wallet Solution (e.g. absence of minimum security or technological requirements)	Wallet Provider
The Wallet Provider's services are unresponsive (e.g. technical errors or lack of connection)	Wallet Provider
The PID Provider or the IT-Wallet ID provider's services are unresponsive (e.g. technical errors)	PID Provider or IT-Wallet ID provider
The Authentication process on the National Identity Provider's service was unsuccessful (e.g. technical errors, unrecognized identity, etc.)	National Identity Provider

Note

When electronic document verification is performed in addition to National Identity Provider authentication, additional error scenarios may occur. For detailed error codes and handling procedures, see *Error Management* (page 474).

5.8.2 Issuance of Electronic Attestations of Attributes Errors

Error type	Actor in charge
The Wallet Instance and/or the PID / IT-Wallet ID are not valid	Wallet Instance
The service for obtaining an Electronic Attestation of Attributes is unavailable (e.g. technical errors)	Electronic Attestations of Attributes Provider, Authentic Source
The User is unable to obtain a specific Electronic Attestation of Attributes in their Wallet Instance (e.g. no eligibility, invalid or expired physical version, etc.)	Authentic Source
The service for obtaining an Electronic Attestation of Attributes cannot be processed within the defined time interval	Electronic Attestations of Attributes Provider
The service for obtaining an Electronic Attestation of Attributes cannot be completed synchronously. The User is requested to wait until the Attestation becomes available (deferred flow)	Electronic Attestations of Attributes Provider, Authentic Source
The service for obtaining an Electronic Attestation of Attributes cannot be processed as the request exceeds the permitted limit (e.g. multiple Attestations requested at once)	Electronic Attestations of Attributes Provider

5.8.3 Presentation of Electronic Attestations Errors

Error type	Actor in charge
The User does not hold the required Attributes contained in one or more Electronic Attestations within their Wallet Instance to access a specific service	Wallet Instance
The Wallet Provider's services or the Relying Party's services are unresponsive (e. g. technical errors or lack of connection)	Wallet Provider, Relying Party

5.8.4 Management of Electronic Attestations Errors

Error type	Actor in charge
The service for revocation/ backup/ restore of an Electronic Attestation of Attributes is unavailable (e.g. technical errors)	Electronic Attestations of Attributes Provider
The service for revocation of PID is unavailable (e.g. technical errors)	PID Provider
The service for erasing information sent to a Relying Party is unavailable (e.g. technical errors)	Relying Party

5.8.5 Deactivation of the Wallet Instance Errors

Error type	Actor in charge
The service for deactivating the Wallet Instance is unavailable (e.g. technical errors)	Wallet Provider

In addition to error management, all Primary Actors MUST also deal with negative feedback resulting from the User's decision to abandon or cancel a flow (e.g. Activation, Acquisition, Presentation, etc.). In such cases, feedback MUST be provided to confirm the User's choice, and it MAY include a Call to Action to continue.

5.9 User Assistance

For effective error management and the resolution of any other issues, Primary Actors MUST ensure adequate support to the User by structuring a simple and effective assistance model based on the following principles:

- **Self-resolution**: to allow the User to consult the frequently asked questions (FAQs) about the content and functionalities of the Wallet Instance, in order to resolve any error cases or issues independently.

- **Guided issue reporting**: to guide the User through the process of opening an assistance ticket, in order to clearly define the issue and facilitate its resolution.
- **Collaboration between actors**: to allow the coordination among each actor involved (Wallet Provider, Electronic Attestations of Attributes Provider, PID Provider, and Authentic Source), according to their specific roles and operational procedures.
- **Efficient communication**: to allow the User to track the updated status of their request throughout all stages of processing, with clear, continuous, and coordinated communication.

To apply these best practices, the involved actors SHOULD implement the following hierarchical support levels:

1. **Level I | Self-management**: the Wallet Provider SHOULD allow the User to access to a Frequently Asked Questions (FAQ) section within their Wallet Instance to clarify doubts and resolve certain issues independently. Each actor SHOULD create specific FAQs and corresponding answers regarding the data and functionalities they provide to the Wallet Provider or in their Touchpoints. For certain error cases, the Wallet Provider SHOULD provide another actor's direct channel of support to facilitate timely management and avoid opening an assistance request within the Wallet Instance.
2. **Level II | Requesting assistance** from the Wallet Provider: if Level I is insufficient, the Wallet Provider SHOULD give the User the possibility to open one or more assistance requests, perform a diagnosis and proceed with resolving the issue, if within their competence. These requests SHOULD be managed through the Wallet Instance or other Wallet Provider Touchpoints. The Wallet Provider MUST diagnose and resolve the issue if it falls under their responsibility.
3. **Level III | Forwarding the request to the responsible actor**: if Level II is insufficient, the Wallet Provider SHOULD ensure that the request is forwarded to the responsible actor (Electronic Attestations of Attributes Provider, PID Provider, or Authentic Source), who ensures the availability of dedicated back-office channels to resolve the issue and communicate the outcome to the User.

Below are the User Experience requirements that the Wallet Solution Provider MUST guarantee through their own Wallet Solution:

- The User accesses to assistance options at any point during the User Experience, with a clear indication of how to access them;
- The User opens an assistance request through their Wallet Instance or other Touchpoints provided by the Wallet Provider;
- When a support request is open, the User receives prompt confirmation that the request has been acknowledged;
- The User is informed in advance if it is necessary to present their data with third parties;

- The User is informed when an assistance request needs to be managed outside of their Wallet Instance, such as on third-party channels;
- The User tracks the status of the request at any time through functionalities that MUST be made available by the actors dealing with the request.

5.10 User Feedback

User feedback collection allows for monitoring the User Experience, identifying potential areas for optimization, and continuously measuring the effectiveness of the service. Each Wallet Provider SHOULD establish a structured feedback collection system to monitor and improve the User Experience.

This feedback system MAY be fed by two different types of feedback collection:

- **Transactional Feedback** (Customer Effort Score, Customer Satisfaction): collected in response to specific actions, such as adding an Electronic Attestation or completing a presentation and verification process;
- **Relational Feedback** (Net Promoter Score): not in connection with specific actions, collected to measure the overall perception of the User in terms of satisfaction, loyalty, and likelihood of recommending the service to third-party users.

Here are some suggestions for implementing these types of feedback tools:

Transactional Feedback Collection

- **Customer Effort Score (CES)**: To measure the ease of use of the functionalities, surveys MAY be provided, for example, through components like modals or pop-ups within the Wallet Instance, triggered at the conclusion of specific actions or processes. Examples include:
  - After completing the process of obtaining an Electronic Attestation;
  - After the Authentication process, if positive;
  - After the presentation process, particularly at the end of the first presentation opportunity and no more than once every 6 months;
  - After the revocation and deactivation processes, to explore the reasons behind these actions.
- **Customer Satisfaction Survey (CSAT)**: To measure the overall satisfaction of the User after a prolonged period of using the Wallet Instance, surveys MAY be provided, for example, through modals or pop-ups within the Wallet Instance. It is recommended to use the CSAT at intervals of no less than six months and as an alternative to CES, to avoid overwhelming Users with too frequent surveys.

Relational Feedback Collection

- **Net Promoter Score (NPS)**: To measure User loyalty and the likelihood of recommending the service to others, an evaluation MAY be requested from the User once or twice a year, either through the same service delivery channel (e.g. the Wallet Instance) or external channels such as email or SMS. This should be aligned with the overall feedback collection strategy.

CHAPTER

SIX

THE INFRASTRUCTURE OF TRUST

The IT-Wallet ecosystem operates within a federated trust infrastructure where participating entities establish cryptographic trust relationships and maintain compliance with common security standards. This infrastructure provides the foundation for secure Digital Credential operations across the ecosystem participants.

This section outlines the implementation of the Trust Model in an infrastructure that complies with OpenID Federation 1.0 OID-FED. This infrastructure involves a RESTful API for distributing metadata, metadata policies, trust marks, cryptographic public keys and X.509 Certificates, and the revocation status of the participants, also called Federation Entities.

The national infrastructure involves a RESTful API for distributing metadata, metadata policies, trust marks, cryptographic public keys and X.509 Certificates, and the revocation status of the participants, also called Federation Entities.

This trust infrastructure works in coordination with the Registry Infrastructure (see *Registry Infrastructure* (page 149)) to enable the entity onboarding processes detailed in *Entity Onboarding* (page 225). In particular, it enables the technical implementation of the onboarding processes described in *Entity Onboarding* (page 225) and supports the operational scenarios illustrated in *Onboarding Journey Maps* (page 29).

The Trust Infrastructure provides the cryptographic mechanisms that allow new entities (Credential Issuers, Relying Parties, Wallet Providers) to establish verifiable trust relationships during their registration process. Without this infrastructure, entities would not be able to prove their compliance status or operational capabilities to other ecosystem participants.

Throughout an entity's operational lifecycle, the Trust Infrastructure maintains up-to-date trust attestations, handles key rotation, manages revocation scenarios, and supports compliance monitoring. This directly supports the lifecycle management procedures detailed in *Entity Onboarding* (page 225).

In this representation, both the *National Trust Anchor* and the Intermediates assume the role of Registration Authority.

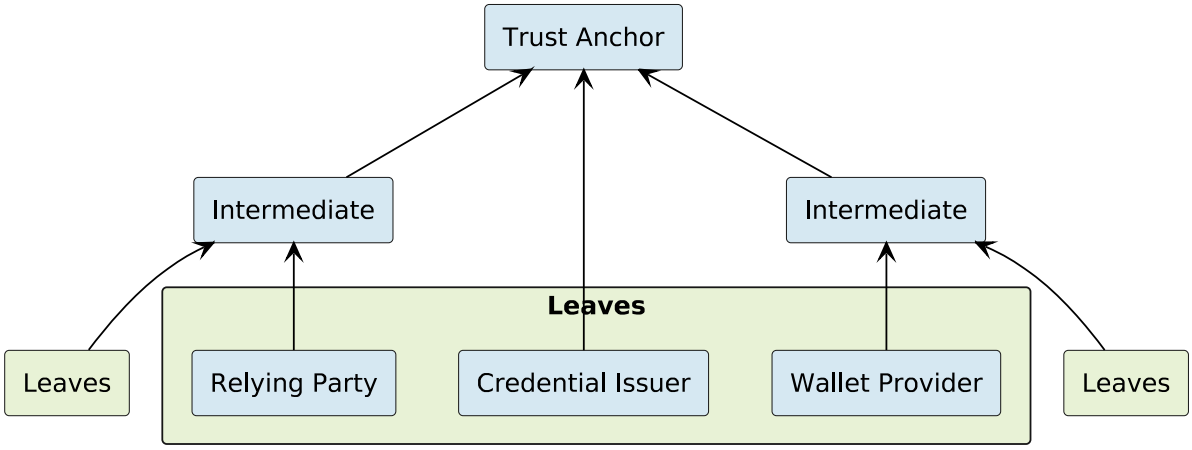


Fig. 6.1: The roles within the Federation, where the Trust Anchor oversees its subordinates, which include one or more Intermediates and Leaves.

6.1 Federation Roles

All the participants are Federation Entities that MUST be registered by a Registration Body, except for Wallet Instances which are End-User's personal devices authenticated by their Wallet Provider.

Note

The Wallet Instance, as a personal device, is deemed reliable through a verifiable attestation issued and signed by a trusted third party.

This is called *Wallet Instance Attestation* and is documented in the dedicated *Wallet Instance Attestation Issuance* (page 285).

Role in Onboarding: During entity registration, the Trust Anchor and Intermediates act as Federation Authorities. This establishes the participant's position in the trust hierarchy and enables them to participate in credential operations. Leaves (Credential Issuers, Relying Parties, Wallet Providers) undergo registration to prove their eligibility and receive authorization to perform their designated functions.

Role in Operations: During Credential issuance and presentation, these roles enable distributed trust validation without requiring centralized verification for each transaction. Leaves utilize their registered status to issue Credentials, verify presentations, or provide Wallet services to end users.

Below the table with the summary of the Federation Entity roles, mapped on the corresponding EUDI Wallet roles.

EUDI Role	Federation Role	Notes
Public Key Infrastructure (PKI)	*National Trust Anchor*	The Federation has PKI capabilities and uses OpenID Federation 1.0 OID-FED. The Entity that configures the entire infrastructure at national level is the *National Trust Anchor*.
Qualified Trust Service Provider (QTSP)	Leaf	
Person Identification Data Provider	Leaf	
Qualified Electronic Attestations of Attributes Provider	Leaf	
Electronic Attestations of Attributes Provider	Leaf	
Relying Party	Leaf	
Trust Service Provider (TSP)	Leaf	
Wallet Provider	Leaf	

6.2 Trust Infrastructure and Registry Integration

The Trust Infrastructure implements the Federation Registry component of the Registry Infrastructure. The Federation Registry maintains the authoritative list of trusted entities through the federation endpoints defined in this section, including entity listing (/list), subordinate statements (/fetch), trust mark validation (/trust_mark_status), subordinate events (/federation_subordinate_events_endpoint), and historical key management (/historical-jwks).

This Federation Registry operates alongside other registry components (Claims Registry, AS Registry, Digital Credentials Catalog, Taxonomy) to provide comprehensive ecosystem support. For complete registry architecture and component interactions, see *Registry Infrastructure* (page 149).

6.3 General Properties

The architecture of the trust infrastructure is built upon the following core principles:

Identifier	Property	Description
P1	**Security**	Incorporates mechanisms to ensure the integrity, confidentiality, and authenticity of the Trust Relationships and interactions within the federation.
P2	**Privacy**	Designed to respect and protect the privacy of the entities involved, minimal disclosure is part of this.
P3	**Interoperability**	Supports seamless interaction and trust establishment between diverse systems and entities within the federation.
P4	**Transitive Trust**	Trust established indirectly through a chain of trusted relationships, enabling entities to trust each other based on common authorities and trusted intermediaries.
P5	**Delegation**	Technical ability or feature to delegate authority or responsibilities to other entities, allowing for a distributed trust mechanism.
P6	**Scalability**	Designed to efficiently manage an increasing number of entities or interactions without a significant increase in trust management complexity.
P7	**Flexibility**	Adaptable to various operational and organizational needs, allowing entities to define and adjust their Trust Relationships and policies.
P8	**Autonomy**	While part of a federated ecosystem, each entity retains control over its own definitions and configurations.
P9	**Decentralization**	Unlike traditional centralized systems, the trust infrastructure should allow a decentralized approach.

6.4 Trust Infrastructure Requirements

This section includes the requirements necessary for the successful implementation and operation of the infrastructure of trust.

Table 6.3: Functional Requirements

ID	Description
FR1	**Federation Trust Establishment**: the system must be able to establish trust between different entities (Credential Issuers, Relying Parties, etc.) within a federation, using cryptographic signatures for secure information exchange about the participants in the ecosystem.
FR2	**Entity Authentication**: the system must implement mechanisms for authenticating entities within the federation, ensuring compliance with the shared rules.
FR3	**Signature Validation**: the system must support the creation, verification, and validation of electronic signatures and provide standard and secure mechanisms to obtain the cryptographic public keys required for the signature validation.
FR4	**Time Stamping**: the signed artifacts must contain time stamps to ensure the integrity and non-repudiation of transactions over time, thanks to the interfaces, services, storage model and approaches defined within the federation.

continues on next page

Table 6.3 – continued from previous page

ID	Description
FR5	**Certificate Validation**: the system requires confidential transmission, secured via TLS over HTTP, and validation of certificates for website authentication.
FR6	**Interoperability and Standards Compliance**: ensure interoperability between federation members by adhering to technical standards, facilitating cross-border electronic transactions.
FR7	**Data Protection and Privacy**: implement data protection measures in compliance with GDPR regulations, ensuring the privacy and security of personal data processed within the federation.
FR8	**Dispute Resolution and Liability**: establish clear procedures for dispute resolution and define liability among federation members.
FR9	**Emergency and Revocation Services**: implement mechanisms for the immediate revocation of participants in case of security breaches or other emergencies.
FR10	**Scalable Trust Infrastructure**: the system must support scalable trust establishment mechanisms, leveraging approaches and technical solutions that complement delegation transitive approaches to efficiently manage Trust Relationships as the federation grows, removing central registries that might technically or administratively fail.
FR11	**Efficient Storage Scalability**: implement a storage solution that scales horizontally to accommodate increasing data volumes while minimizing central storage and administrative costs. The system should enable members to independently store and present historical trust attestations and signed artifacts during dispute resolutions, with the federation infrastructure maintaining only a registry of historical keys to validate the historical data, stored and provided by the participants.
FR12	**Verifiable Attestation (Trust Mark)**: incorporate a mechanism for issuing and verifying verifiable attestations that serve as proof of compliance with specific profiles or standards. This allows entities within the federation to demonstrate adherence to agreed-upon security, privacy, and operational standards.
FR13	**Decentralized Dispute Resolution Mechanism**: design a decentralized mechanism for dispute resolution that allows federation members to independently verify historical trust establishment and signed artifacts, reducing reliance on central authorities and streamlining the resolution process.
FR14	**Cross-Federation Interoperability**: ensure the system is capable of interoperating with other federations or Trust Frameworks, facilitating cross-federation transactions and trust establishment without compromising security or compliance.
FR15	**Autonomous Registration Bodies**: the system must facilitate the integration of autonomous registration bodies that operate in compliance with federation rules. These bodies are tasked with evaluating and registering entities within the federation, according to the pre-established rules and their compliance that must be periodically asserted.
FR16	**Periodic Auditing of Registration Bodies and Entities**: implement mechanisms for the periodic auditing and monitoring of the compliance status of both registration bodies and their registered entities.

continues on next page

Table 6.3 – continued from previous page

ID	Description
FR17	**Attestation of Compliance for Personal Devices**: trusted bodies, in the form of federation entities, should issue attestations of compliance and provide signed proof of such compliance for the hardware of personal devices used within the federation. These attestations should be attested and periodically renewed to ensure the devices meet current security standards.
FR18	**Automated Compliance Monitoring**: the system should include automated tools for monitoring the compliance of entities with federation standards. This automation aids in the early detection of potential compliance issues.
FR19	**Secure Protocol Capabilities Binding**: the secure protocol must enable the exchange of protocol-specific capabilities data as cryptographically-bound metadata attached to a specific identity. This metadata should define the technical capabilities associated with the identity, ensuring verifiable proof and tamper-proof association for robust trust establishment and access control.

6.5 Federation API endpoints

OpenID Federation 1.0 uses RESTful Web Services secured over HTTPs. OpenID Federation 1.0 defines which are the web endpoints that the participants MUST make publicly available. The table below summarises the endpoints and their scopes.

All the endpoints listed below are defined in the OID-FED specs.

endpoint name	http request	scope	required for
federation metadata	**GET** .well-known/openid-federation	Metadata that an Entity publishes about itself, verifiable with a trusted third party (Superior Entity). It's called Entity Configuration.	Trust Anchor, Intermediate, Wallet Provider, Relying Party, Credential Issuer
subordinate list endpoint	**GET** /list	Lists the Subordinates. See OID-FED Section 8.2	Trust Anchor, Intermediate
fetch endpoint	**GET** /federation_fetch_endpoint?sub=https://rp.example.org	Returns a signed JWT about a specific subject, its Subordinate. It's called Subordinate Statement. See OID-FED Section 8.1	Trust Anchor, Intermediate

continues on next page

Table 6.4 – continued from previous page

endpoint name	http request	scope	required for
trust mark status	**POST** /trust_mark_status	Returns the status of the issuance (validity) of a Trust Mark related to a specific subject. See OID-FED Section 8.4	Trust Anchor, Intermediate
trust mark list	**GET** /trust_mark_listing?trust_mark_type=...	Lists all entities for which Trust Marks have been issued and are still valid. See OID-FED Section 8.5	Trust Anchor, Intermediate
trust mark	**GET** /trust_mark?trust_mark_type=...	Returns the Trust Mark related to a specific subject. See OID-FED Section 8.6	Trust Anchor, Intermediate
historical keys	**GET** /federation_historical_keys	Lists the expired and revoked keys, with the motivation of the revocation. See OID-FED Section 8.7	Trust Anchor, Intermediate
subordinate events	**GET** /federation_subordinate_events_endpoint?sub=https://rp.example.org	Returns a historical track of registration events about Immediate Subordinates, such as registration, revocation, and updates of their Federation Entity Keys. See the section *Federation Subordinate Events Endpoint* (page 146) for more details.	Trust Anchor, Intermediate

All the responses of the federation endpoints are in the form of signed JWT, with the exception of the Subordinate Listing endpoint and the Trust Mark Status endpoint that are served as plain JSON by default. The Federation Subordinate Events Endpoint also returns signed JWTs with the content type `application/entity-events-statement+jwt`.

6.6 Configuration of the Federation

The configuration of the federation is published by the Trust Anchor within its Entity Configuration, it is available at the well-known web path corresponding to **.well-known/openid-federation**.

All the participants in the federation MUST obtain the federation configuration before entering the operational phase, and they MUST keep it up-to-date. The federation configuration is the Trust Anchor's Entity Configuration, it contains the public keys for signature operations.

Below is a non-normative example of a Trust Anchor Entity Configuration, where each parameter is documented in the OpenID Federation specification:

```
{
    "alg": "ES256",
    "kid": "FifYx03bnosD8m6gYQIfNHNP9cM_Sam9Tc5nLloIIrc",
    "typ": "entity-statement+jwt"
}
```

```
{
    "exp": 1649375259,
    "iat": 1649373279,
    "iss": "https://trust-anchor.eid-wallet.example.it",
    "sub": "https://trust-anchor.eid-wallet.example.it",
    "jwks": {
        "keys": [
            {

                "kty": "EC",
                "kid": "X2ZOMHNGSDc4ZlBrcXhMT3MzRmRZOG9Jd3o2QjZDam51cUhhUFRuOWd0WQ",
                "crv": "P-256",
                "x": "1kNR9Ar3MzMokYTY8BRvRIue85NIXrYX4XD3K4JW7vI",
                "y": "slT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrY",
                "x5c": [
                  // <self-issued X.509 certificate about the Trust Anchor>
                  ]
            }
        ]
    },
    "metadata": {
        "federation_entity": {
            "organization_name": "example TA",
            "contacts":[
                "tech@eid.trust-anchor.example.eu"
            ],
            "homepage_uri": "https://trust-anchor.eid-wallet.example.it",
            "logo_uri":"https://trust-anchor.eid-wallet.example.it/static/svg/logo.svg",
```

```
            "federation_fetch_endpoint": "https://trust-anchor.eid-wallet.example.it/fetch",
            "federation_resolve_endpoint": "https://trust-anchor.eid-wallet.example.it/resolve"
,
            "federation_list_endpoint": "https://trust-anchor.eid-wallet.example.it/list",
            "federation_trust_mark_status_endpoint": "https://trust-anchor.eid-wallet.example.i
t/trust_mark_status",
            "federation_trust_mark_list_endpoint": "https://trust-anchor.eid-wallet.example.it/
trust_mark_listing",
            "federation_subordinate_events_endpoint": "https://trust-anchor.eid-wallet.example.
it/events"
        }
    },
    "trust_mark_issuers": {
        "https://trust-anchor.eid-wallet.example.it/openid_relying_party/public": [
            "https://trust-anchor.eid-wallet.example.it",
            "https://public.intermediary.other.org"
        ],
        "https://trust-anchor.eid-wallet.example.it/openid_relying_party/private": [
            "https://private.other.intermediary.org"
        ]
    }
}
```


6.7 Entity Configuration

The Entity Configuration is the verifiable document that each Federation Entity MUST publish on its own behalf, in the **.well-known/openid-federation** endpoint.

The Entity Configuration HTTP Response MUST set the media type to *application/entity-statement+jwt*.

The Entity Configuration MUST be cryptographically signed. The public part of this key MUST be provided in the Entity Configuration and within the Subordinate Statement issued by a immediate superior and related to its subordinate Federation Entity.

The Entity Configuration MAY also contain one or more Trust Marks.

Role in Onboarding: New entities publish their Entity Configuration as part of their registration process, declaring their capabilities, supported protocols, and compliance status to the federation. The configuration serves as the entity's initial declaration of its technical readiness and operational scope, enabling other participants to discover and validate its registration status.

Role in Operations: During Credential operations, Entity Configurations are retrieved by wallets, credential issuers, and relying parties to verify the current operational status, supported capabilities, and compliance attestations of other entities. This enables dynamic discovery of

service endpoints, cryptographic keys, and protocol versions required for secure Credential exchange.

Technical details about Entity Configuration of Wallet Provider, Credential Issuer and Relying Party are given in Section *Wallet Provider Entity Configuration* (page 305), credential-issuer-entity-configuration:Credential Issuer Entity Configuration and *Relying Party Entity Configuration* (page 352) respectively.

Note

Entity Configuration Signature

All the signature-check operations regarding the Entity Configurations, Subordinate Statements and Trust Marks, are carried out with the Federation public keys. For the supported algorithms refer to Section *Cryptografic Algorithm*.

6.7.1 Entity Configurations Common Parameters

The Entity Configurations of all the participants in the federation MUST have in common the parameters listed below.

Claim	Description
iss	String. Identifier of the issuing Entity.
sub	String. Identifier of the Entity to which it is referred. It MUST be equal to `iss`.
iat	UNIX Timestamp with the time of generation of the JWT, coded as NumericDate as indicated at **RFC 7519**.
exp	UNIX Timestamp with the expiry time of the JWT, coded as NumericDate as indicated at **RFC 7519**.
jwks	A JSON Web Key Set (JWKS) **RFC 7517** that represents the public part of the signing keys of the Entity at issue. Each JWK in the JWK set MUST have a key ID (claim kid) and MAY have a *x5c* parameter, as defined in **RFC 7517**. It contains the Federation Entity Keys required for the operations of Trust Evaluation. **x5c**: The *x5c* parameter included in Entity Configuration's *jwks* parameter MUST only contain the self-issued X.509 Certificate about the corresponding *jwk*.
metadata	JSON Object. Each key of the JSON Object represents a metadata type identifier containing JSON Object representing the metadata, according to the metadata schema of that type. An Entity Configuration MAY contain more metadata statements, but only one for each type of metadata (<**entity_type**>). the metadata types are defined in the section Metadata Types.

6.7.2 Entity Configuration Trust Anchor

The Trust Anchor Entity Configuration, in addition to the common parameters listed above, uses the following parameters:

Claim	Description	Required
trust_mark_ issuers	JSON Array that defines which Federation authorities are considered trustworthy for issuing specific Trust Marks, assigned with their unique identifiers.	✓

6.7.3 Trust Marks for Relying Party Intermediaries

In the IT-Wallet ecosystem, Relying Party Intermediaries acting on behalf of Relying Parties (pursuant to Art. 5b(8) of the eIDAS2 Regulation EU_2024_1183) are subject to a specific Trust Mark regime that allows the Wallet to recognize and display to the User that the requesting RP operates through a Relying Party Intermediary.

Trust Mark Issuance to the Relying Party Intermediary

The Trust Anchor MUST include Relying Party Intermediaries in the `trust_mark_issuers` attribute of its own Entity Configuration, authorizing them to issue Trust Marks for their subordinate Leaves. The Trust Anchor issues a Trust Mark to the Relying Party Intermediary attesting its role, using `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier_intermediary` as `trust_mark_type` to distinguish it from the other entities in the ecosystem.

The Relying Party Intermediary MUST include this Trust Mark in its own Entity Configuration as proof of its recognized role within the Federation.

Trust Mark Issuance to Leaves Affiliated with the Relying Party Intermediary

Trust Marks issued to Leaves (Relying Parties) affiliated with a Relying Party Intermediary MUST be issued by the Relying Party Intermediary itself, not directly by the Trust Anchor.

A Relying Party affiliated with a Relying Party Intermediary MUST include in its own Entity Configuration:

- The Trust Mark issued by the Relying Party Intermediary, as proof of the successful onboarding process managed by the Relying Party Intermediary;
- Its own `authority_hints` containing the URL of the Relying Party Intermediary (not directly of the Trust Anchor), thus indicating the correct hierarchical chain.

Below is a non-normative example of a Trust Anchor Entity Configuration showing the updated `trust_mark_issuers` configuration with authorized Relying Party Intermediaries:

```
{
  "trust_mark_issuers": {
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_verifier": [
      "https://trust-anchor.eid-wallet.example.it",
      "https://intermediary.example.org"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_verifier_intermediary": [
      "https://trust-anchor.eid-wallet.example.it"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_issuer": [
      "https://trust-anchor.eid-wallet.example.it"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_wallet_pro
vider": [
      "https://trust-anchor.eid-wallet.example.it"
    ]
  }
}
```

Note

The Trust Mark with identifier ending with `federation-entity/openid_credential_verifier_intermediary` is issued exclusively by the Trust Anchor and is the one the Wallet uses to verify that an entity is a recognized Relying Party Intermediary. The `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier` Trust Marks SHOULD also be issued by Relying Party Intermediaries for their affiliated Leaves.

6.7.4 Entity Configuration Leaves and Intermediates

In addition to the previously defined claims, the Entity Configuration of the Leaves and of the Intermediate Entities uses the following parameters:

Claim	Description	Required
authority_hints	Array of URLs (String). It contains a list of URLs of the immediate superior entities, such as the Trust Anchor or an Intermediate, that issues an Subordinate Statement related to this subject.	✓
trust_marks	A JSON Array containing the Trust Marks.	✓

6.7.5 Metadata Types

In this section are defined the main metadata types mapped to the roles of the ecosystem, giving the references of the metadata protocol for each of these.

Note

The entries that don't have any reference to a known draft or standard are intended to be defined in this technical reference.

OpenID Entity	EUDI Entity	Metadata Type	References
Trust Anchor	Trust Anchor	`federation_entity`	OID-FED
Intermediate	Intermediate	`federation_entity`	OID-FED
Wallet Provider	Wallet Provider	`federation_entity`, `openid_wallet_provider`	OID-FED-WALLET
Authorization Server		`federation_entity`, `oauth_authorization_server`	OPENID4VCI
Credential Issuer	PID Provider, (Q)EAA Provider	`federation_entity`, `openid_credential_issuer`, [`oauth_authorization_server`]	OPENID4VCI
Relying Party	Relying Party	`federation_entity`, `openid_credential_verifier`	OID-FED, OpenID4VP

Note

Wallet Provider metadata is defined in the section below.

Wallet Solution (page 269).

Note

In instances where a PID/EAA Provider implements both the Credential Issuer and the Authorization Server, it MUST incorporate both `oauth_authorization_server` and `openid_credential_issuer` within its metadata types. Other implementations may divide the Credential Issuer from the Authorization Server, when this happens the Credential Issuer metadata MUST contain the *authorization_servers* parameters, including the Authorization Server unique identifier. Furthermore, should there be a necessity for User Authentication by the Credential Issuer, it could be necessary to include the relevant metadata type, either `openid_relying_party` or `openid_credential_verifier`.

6.8 Metadata of federation_entity Leaves

The *federation_entity* metadata for Leaves contain the following claims.

Claim	Description
organization_name	REQUIRED. See OID-FED Section 5.2.2
homepage_uri	REQUIRED. REQUIRED if `organization_uri` is absent.
organization_uri	CONDITIONAL. REQUIRED if `homepage_uri` is absent. See OID-FED Section 5.2.2
policy_uri	REQUIRED. See OID-FED Section 5.2.2
logo_uri	REQUIRED. URL of the entity's logo; it MUST be in SVG format. See OID-FED Section 5.2.2
contacts	REQUIRED. Institutional verified email address (PEC) of the entity. See OID-FED Section 5.2.2
federation_resolve_endpoint	OPTIONAL. See OID-FED Section 8.3
tos_uri	OPTIONAL. URL string that points to a human-readable terms of service document for the client that describes a contractual relationship between the end-user and the client that the end-user accepts when authorizing the client. See OID-FED.

6.9 Subordinate Statements

Trust Anchors and Intermediates publish Subordinate Statements related to their immediate Subordinates. The Subordinate Statement MAY contain a metadata policy and the Trust Marks related to a Subordinate.

The metadata policy, when applied, makes one or more changes to the final metadata of the Leaf. The final metadata of a Leaf is derived from the Trust Chain that contains all the statements, starting from the Entity Configuration up to the Subordinate Statement issued by the Trust Anchor.

Trust Anchors and Intermediates MUST expose the Federation Fetch endpoint, where the Subordinate Statements are requested to validate the Leaf's Entity Configuration signature.

Note

The Federation Fetch endpoint MAY also publish X.509 Certificates for each of the public keys of the Subordinate. Making the distribution of the issued X.509 Certificates via a RESTful service.

Role in Onboarding: During entity registration, Trust Anchors and Intermediates issue Subordinate Statements to formally attest the registration and capabilities of new entities. These statements establish the hierarchical trust relationship and apply any required metadata policies that constrain or enhance the entity's declared capabilities based on federation policies.

Role in Operations: During Credential operations, Subordinate Statements are retrieved to validate Trust Chains and apply current metadata policies. They enable real-time verification of an entity's registration status and ensure that operational capabilities comply with federation-wide policies and the entity's authorized scope.

Below there is a non-normative example of an Subordinate Statement issued by an Registration Body (such as the Trust Anchor or its Intermediate) in relation to one of its Subordinates.

```
{
    "alg": "ES256",
    "kid": "em3cmnZgHIYFsQ090N6B3Op7LAAqj8rghMhxGmJstqg",
    "typ": "entity-statement+jwt"
}

{
    "exp": 1649623546,
    "iat": 1649450746,
    "iss": "https://intermediate.example.org",
    "sub": "https://rp.example.it",
    "jwks": {
        "keys": [ // keys about the Subordinate
```

```
            {
                "kty": "EC",
                "kid": "2HnoFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "crv": "P-256",
                "x": "1kNR9Ar3MzMokYTY8BRvRIue85NIXrYX4XD3K4JW7vI",
                "y": "slT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrY",
                "x5c": [
                  // <X.509 certificate about the Subordinate>
                  ]
            }
        ]
    },
    "metadata_policy": {
        "openid_credential_verifier": {
            "scope": {
                "subset_of": [
                     "eu.europa.ec.eudiw.pid.1",
                     "given_name",
                     "family_name",
                     "email"
                  ]
            },
            "vp_formats": {
                "dc+sd-jwt": {
                    "sd-jwt_alg_values": [
                        "ES256",
                        "ES384"
                    ],
                    "kb-jwt_alg_values": [
                        "ES256",
                        "ES384"
                    ]
                }
            }
        }
     }
}
```

Note

Subordinate Statement Signature

The same considerations and requirements made for the Entity Configuration and in relation to the signature mechanisms MUST be applied for the Subordinate Statements.

6.9.1 Subordinate Statement

The Subordinate Statement issued by Trust Anchors and Intermediates contains the following attributes:

Claim	Description	Required
iss	See OID-FED Section 3 for further details.	✓
sub	See OID-FED Section 3 for further details.	✓
iat	See OID-FED Section 3 for further details.	✓
exp	See OID-FED Section 3 for further details.	✓
jwks	Federation JWKS of the *sub* entity. See OID-FED Section 3 for further details.	✓
metadata_policy	JSON Object that describes the Metadata policy. Each key of the JSON Object represents an identifier of the metadata type and each value MUST be a JSON Object that represents the metadata policy according to that metadata type. Please refer to the OID-FED specifications, Section 6.1, for the implementation details.	✗
trust_marks	JSON Array containing the Trust Marks issued by itself for the subordinate subject.	✗
constraints	It MAY contain the **allowed_leaf_entity_types**, that restricts what types of metadata the subject is allowed to publish. It MAY contain the maximum number of Intermediates allowed between a itself and the Leaf (**max_path_length**)	✓

6.10 Federation Discovery

Federation Entity Discovery is the process by which participants verify the identity and status of other federation entities. This involves querying federation endpoints to confirm entity validity status and compliance with the Trust Framework.

The discovery process uses the Federation API to collect entity information and produces Trust Chains that validate entity compliance and authorization within the federation.

The discovery process establishes the foundational concepts that are then applied in specific operational scenarios as detailed in the Trust Evaluation Mechanism section.

Note

Trust Anchors MUST distribute their Federation Public Keys through secure out-of-band mechanisms, such as publishing them on a verified web page or storing them in a remote repository as part of a trust list. The rationale behind this requirement is that relying solely on the data provided within the Trust Anchor's Entity Configuration does not adequately mitigate risks associated with DNS and TLS manipulation attacks. To ensure security, all participants MUST obtain the Trust Anchor's public keys using these out-of-band methods. They should then compare these keys with those obtained from the Trust Anchor's Entity Configuration, discarding any keys that do not match. This process helps to ensure the integrity and authenticity of the Trust Anchor's public keys and the overall security of the federation (*WP_017* (page 739)).

Each Subordinate Statement is verifiable over time and MUST have an expiration date. The revocation of each statement is verifiable in real time and online (only for remote flows) through the federation endpoints.

Note

The revocation of an Entity is made with the unavailability of the Subordinate Statement related to it. If the Trust Anchor or its Intermediate doesn't publish a valid Subordinate Statement, or if it publishes an expired/invalid Subordinate Statement, the subject of the Subordinate Statement MUST be intended as not valid or revoked.

The concatenation of the statements, through the combination of these signing mechanisms and the binding of claims and public keys, forms the Trust Chain.

The Trust Chains can also be verified offline, using one of the Trust Anchor's public keys.

Note

Since the Wallet Instance is not a Federation Entity, the Trust Evaluation Mechanism related to it **requires the presentation of the Wallet Attestation during the credential issuance and presentation phases**.

The Wallet Attestation conveys all the required information pertaining to the instance, such as its public key and any other technical or administrative information, without any User's personal data.

6.11 Trust Chain

The Trust Chain is a sequence of verified statements that validates a participant's compliance with the Federation. It has an expiration date time, beyond which it MUST be renewed to obtain the fresh and updated metadata. The expiration date of the Trust Chain is determined by the earliest expiration timestamp among all the expiration timestamp contained in the statements. No Entity can force the expiration date of the Trust Chain to be higher than the one configured by the Trust Anchor.

Role in Onboarding: During the entity onboarding, Trust Chains are constructed to prove the complete hierarchical trust relationship from the Trust Anchor to the new entity.

Role in Operations: During Credential issuance and presentation, Trust Chains provide cryptographic proof of entity validity and compliance status.

Below is an abstract representation of a Trust Chain.

```
[
    "EntityConfiguration-as-SignedJWT-selfissued-byLeaf",
    "EntityStatement-as-SignedJWT-issued-byTrustAnchor"
]
```

Below is a non-normative example of a Trust Chain, composed by a JSON Array containing JWTs, with an Intermediate involved.

[
"eyJhbGciOiJFUzI1NiIsImtpZCI6Ik5GTTFXVVZpVWxZelVXcExhbWxmY0VwUFJWWTJWWFpJUmpCblFYWm1SSGhLWVVW
WVVsZFRRbkEyTkEiLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p3dCJ9.eyJleHAiOjE2NDk1OTA2MD
IsImlhdCI6MTY0OTQxNzg2MiwiaXNzIjoiaHR0cHM6Ly9ycC5leGFtcGxlLm9yZyIsInN1YiI6Imh0dHBzOi8vcnAuZXhhb
XBsZS5vcmciLCJqd2tzIjp7ImtleXMiOlt7Imt0eSI6IkVDIiwia2lkIjoiTkZNMVdVVmlVbFl6VVdwTGFtbGZjRXBQUlZZ
MlZYWklSakJuUVhabVJIaEtZVVZZVWxkVFFuQTJOQSIsImNydiI6IlAtMjU2IiwieCI6InVzbEMzd2QtcFgzd3o0YlJZbnd
5M2x6cGJHWkZoTjk2aEwyQUhBM01RNlkiLCJ5IjoiVkxDQlhGV2xkTlNOSXo4a0gyOXZMUjROMThCa3dHT1gyNnpRb3J1UT
FNNCJ9XX0sIm1ldGFkYXRhIjp7Im9wZW5pZF9yZWx5aW5nX3BhcnR5Ijp7ImFwcGxpY2F0aW9uX3R5cGUiOiJ3ZWIiLCJjb
GllbnRfaWQiOiJodHRwczovL3JwLmV4YW1wbGUub3JnLyIsImNsaWVudF9yZWdpc3RyYXRpb25fdHlwZXMiOlsiYXV0b21h
dGljIl0sImp3a3MiOnsia2V5cyI6W3sia3R5IjoiRUMiLCJraWQiOiJORk0xV1VWaVVsWXpVV3BMYW1sZmNFcFBSVlkyVlh
aSVJqQm5RWFptUkhoS1lVVllVbGRUUW5BMk5BIiwiY3J2IjoiUC0yNTYiLCJ4IjoidXNsQzN3ZC1wWDN3ejRiUllud3kzbH
pwYkdaRmhOOTZoTDJBSEEzTVE2WSIsInkiOiJWTENCWEZXbGROU05JejhrSDI5dkxSNE4xOEJrd0dPWDI2elFvcnVRMU00I
n1dfSwiY2xpZW50X25hbWUiOiJOYW1lIG9mIGFuIGV4YW1wbGUgb3JnYW5pemF0aW9uIiwiY29udGFjdHMiOlsib3BzQHJw
LmV4YW1wbGUuaXQiXSwiZ3JhbnRfdHlwZXMiOlsicmVmcmVzaF90b2tlbiIsImF1dGhvcml6YXRpb25fY29kZSJdLCJyZWR
pcmVjdF91cmlzIjpbImh0dHBzOi8vcnAuZXhhbXBsZS5vcmcvb2lkYy9jYWxsYmFjay8iXSwicmVzcG9uc2VfdHlwZX
MiOlsiY29kZSJdLCJzY29wZSI6ImV1LmV1cm9wYS5lYy5ldWRpdy5waWQuMSBldS5ldXJvcGEuZWMuZXVkaXcucGlkLml0L
jEgZW1haWwiLCJzdWJqZWN0X3R5cGUiOiJwYWlyd2lzZSJ9LCJmZWRlcmF0aW9uX2VudGl0eSI6eyJmZWRlcmF0aW9uX3Jl
c29sdmVfZW5kcG9pbnQiOiJodHRwczovL3JwLmV4YW1wbGUub3JnL3Jlc29sdmUvIiwib3JnYW5pemF0aW9uX25hbWUiOiJ
FeGFtcGxlIFJQIiwiaG9tZXBhZ2VfdXJpIjoiaHR0cHM6Ly9ycC5leGFtcGxlLml0IiwicG9saWN5X3VyaSI6Imh0dHBzOi
8vcnAuZXhhbXBsZS5pdC9wb2xpY3kiLCJsb2dvX3VyaSI6Imh0dHBzOi8vcnAuZXhhbXBsZS5pdC9zdGF0aWMvbG9nby5zd
mciLCJjb250YWN0cyI6WyJ0ZWNoQGV4YW1wbGUuaXQiXX19LCJ0cnVzdF9tYXJrcyI6W3siaWQiOiJodHRwczovL3JlZ2lz
dHJ5LmVpZGFzLnRydXN0LWFuY2hvci5leGFtcGxlLmV1L29wZW5pZF9yZWx5aW5nX3BhcnR5L3B1YmxpcyIsICJ0cnVzdF9
tYXJrIjoiZXlKaCBcdTIwMjYifV0sImF1dGhvcml0eV9oaW50cyI6WyJodHRwczovL2ludGVybWVkaWF0ZS5laWRhcy5leG
FtcGxlLm9yZyJdfQ.Un315HdckvhYA-iRregZAmL7pnfjQH2APz82blQO5S0sl1JR0TEFp5E1T913g8GnuwgGtMQUqHPZwV
6BvTLA8g","eyJhbGciOiJFUzI1NiIsImtpZCI6IlNURkRXV2hKY0dWWFgzQjNSVmRaYWtsQ0xUTnVNa000WTNGNlFUTk9k
RXRyZFhGWVlYWjJjWGN0UVEiLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p3dCJ9.eyJleHAiOjE2ND
k2MjM1NDYsImlhdCI6MTY0OTQ1MDc0NiwiaXNzIjoiaHR0cHM6Ly9pbnRlcm1lZGlhdGUuZWlkYXMuZXhhbXBsZS5vcmciL
CJzdWIiOiJodHRwczovL3JwLmV4YW1wbGUub3JnIiwiandrcyI6eyJrZXlzIjpbeyJrdHkiOiJFQyIsImtpZCI6Ik5GTTFX
VVZpVWxZelVXcExhbWxmY0VwUFJWWTJWWFpJUmpCblFYWm1SSGhLWVVWWVVsZFRRbkEyTkEiLCJjcnYiOiJQLTI1NiIsIng
iOiJ1c2xDM3dkLXBYM3d6NGJSWW53eTNsenBiR1pGaE45NmhMMkFIQTNNUTZZIiwieSI6IlZMQ0JYRldsZE5TTkl6OGtIMj
l2TFI0TjE4Qmt3R09YMjZ6UW9ydVExTTQifV19LCJtZXRhZGF0YV9wb2xpY3kiOnsib3BlbmlkX3JlbHlpbmdfcGFydHkiO
nsic2NvcGUiOnsic3Vic2V0X29mIjpbImV1LmV1cm9wYS5lYy5ldWRpdy5waWQuMSwgIGV1LmV1cm9wYS5lYy5ldWRpdy5w
aWQuaXQuMSJdfSwicmVxdWVzdF9hdXRoZW50aWNhdGlvbl9tZXRob2RzX3N1cHBvcnRlZCI6eyJvbmVfb2YiOlsicmVxdWV
zdF9vYmplY3QiXX0sInJlcXVlc3RfYXV0aGVudGljYXRpb25fc2lnbmluZ19hbGdfdmFsdWVzX3N1cHBvcnRlZCI6eyJzdW
JzZXRfb2YiOlsiUlMyNTYiLCJSUzUxMiIsIkVTMjU2IiwiRVM1MTIiLCJQUzI1NiIsIlBTNTEyIl19fX0sInRydXN0X21hc
mtzIjpbeyJpZCI6Imh0dHBzOi8vdHJ1c3QtYW5jaG9yLmV4YW1wbGUuZXUvb3BlbmlkX3JlbHlpbmdfcGFydHkvcHVibGlj
LyIsInRydXN0X21hcmsiOiJleUpoYiBcdTIwMjYifV19._qt5-T6DahP3TuWa_27klE8I9Z_sPK2FtQlKY6pGMPchbSI2aH
XY3aAXDUrObPo4CHtqgg3J2XcrghDFUCFGEQ",

```
  "eyJhbGciOiJFUzI1NiIsImtpZCI6ImVXa3pUbWt0WW5kblZHMWxhMjU1ZDJkQ2RVZERSazQwUWt0WVlVMWFhRFZYT1Ro
bFpHdFdXSGQ1WnciLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p3dCJ9.eyJleHAiOjE2NDk2MjM1ND
YsImlhdCI6MTY0OTQ1MDc0NiwiaXNzIjoiaHR0cHM6Ly90cnVzdC1hbmNob3IuZXhhbXBsZS5ldSIsInN1YiI6Imh0dHBzO
i8vaW50ZXJtZWRpYXRlLmVpZGFzLmV4YW1wbGUub3JnIiwiandrcyI6eyJrZXlzIjpbeyJrdHkiOiJFQyIsImtpZCI6IlNU
RkRXV2hKY0dWWFgzQjNSVmRaYWtsQ0xUTnVNa000WTNGNlFUTk9kRXRyZFhGWVlYWjJjWGN0UVEiLCJjcnYiOiJQLTI1NiI
sIngiOiJyQl9BOGdCUnh5NjhVTkxZRkZLR0ZMR2VmWU5XYmgtSzh1OS1GYlQyZkZJIiwieSI6IlNuWVk2Y3NjZnkxcjBISF
hLTGJuVFZsamFndzhOZzNRUEs2WFVoc2UzdkUifV19LCJ0cnVzdF9tYXJrcyI6W3siaWQiOiJodHRwczovL3RydXN0LWFuY
2hvci5leGFtcGxlLmV1L2ZlZGVyYXRpb25fZW50aXR5L3RoYXQtcHJvZmlsZSIsInRydXN0X21hcmsiOiJleUpoYiBcdTIw
MjYifV19.r3uoi-U0tx0gDFlnDdITbcwZNUpy7M2tnh08jlD-Ej9vMzWMCXOCCuwIn0ZT0jS4M_sHneiG6tLxRqj-htI70g
"
]
```


Note

The entire Trust Chain is verifiable by only possessing the Trust Anchor's public keys.

There are events where keys are unavailable to verify the entire trust chain:

- **Key Change by Credential Issuer**: The Credential Issuer MAY update its cryptographic keys. The cryptographic keys MUST be considered valid if evaluated within their originally designated validity period unless a security reason makes them unusable. The revocation reason MUST be published. Historical cryptographic keys, i.e. unused or revoked public cryptographic keys, MUST be published using the Federation Historical Keys Endpoint.
- **Change in Credential Types**: If the Credential Issuer changes the Credential **types** issued, for instance deciding not to issue anymore one or more Credential types, the related public cryptographic keys MUST be available for the originally designated validity period.
- **Credential Issuers Merge**: If a Credential Issuer merges with another, creating a new Organizational Entity or working on behalf of another one using a different hostname or domain, the **previously** available federation configuration and historical keys MUST be kept available at the original Credential Issuer's well-known endpoints.
- **Credential Issuer Becomes Inactive**: If a Credential Issuer becomes inactive, its **related** Entity Configuration and Federation Historical Entity Endpoint MUST be kept available.

6.11.1 Offline Trust Attestation Mechanisms

The offline flows do not allow for real-time evaluation of an Entity's status, such as its revocation. At the same time, using short-lived Trust Chains enables the attainment of trust attestations compatible with the required revocation administrative protocols (e.g., a revocation must be propagated in less than 24 hours, thus the Trust Chain must not be valid for more than that period).

Offline Wallet Trust Attestation

Given that the Wallet Instance cannot publish its metadata online at the *.well-known/openid-federation* endpoint, it MUST obtain a Wallet Attestation issued by its Wallet Provider. The Wallet Attestation MUST contain all the relevant information regarding the security capabilities of the Wallet Instance and its protocol related configuration. It SHOULD contain the Trust Chain related to its issuer (Wallet Provider).

Offline Relying Party Metadata

Since the Federation Entity Discovery is only applicable in online scenarios, it is possible to include the Trust Chain in the presentation requests that the Relying Party may issue for a Wallet Instance.

The Relying Party MUST sign the presentation request, the request SHOULD include the *trust_chain* claim in its JWT header parameters, containing the Federation Trust Chain related to itself.

The Wallet Instance that verifies the request issued by the Relying Party MUST use the Trust Anchor's public keys to validate the entire Trust Chain related to the Relying Party before attesting its reliability.

Furthermore, the Wallet Instance applies the metadata policy, if any.

6.11.2 Trust Chain Fast Renewal

The Trust Chain fast renewal method offers a streamlined way to maintain the validity of a trust chain without undergoing the full discovery process again. It's particularly useful for quickly updating Trust Relationships when minor changes occur or when the Trust Chain is close to expiration but the overall structure of the federation hasn't changed significantly.

The Trust Chain fast renewal process is initiated by fetching the leaf's Entity Configuration anew. However, unlike the federation discovery process that may involve fetching Entity Configurations starting from the authority hints, the fast renewal focuses on directly obtaining the

Subordinate Statements. These statements are requested using the *source_endpoint* provided within them, which points to the location where the statements can be fetched.

6.12 Trust Evaluation Mechanism

Trust Anchors MUST distribute their Federation Public Keys through secure out-of-band mechanisms, such as publishing them on a verified web page or storing them in a remote repository as part of a trust list. The rationale behind this requirement is that relying solely on the data provided within the Trust Anchor's Entity Configuration does not adequately mitigate risks associated with DNS and TLS manipulation attacks. To ensure security, all participants MUST obtain the Trust Anchor's public keys using these out-of-band methods. They should then compare these keys with those obtained from the Trust Anchor's Entity Configuration, discarding any keys that do not match. This process helps to ensure the integrity and authenticity of the Trust Anchor's public keys and the overall security of the federation.

The Trust Anchor publishes the list of its Subordinates (Federation Subordinate Listing endpoint), the attestations of their metadata and public keys (Subordinate Statements), and historical events related to their lifecycle (Federation Subordinate Events endpoint).

Each participant, including Trust Anchor, Intermediate, Credential Issuer, Wallet Provider, and Relying Party, publishes its own metadata and public keys (Entity Configuration endpoint) in the well-known web resource **.well-known/openid-federation**.

Each of these can be verified using the Subordinate Statement issued by a superior, such as the Trust Anchor or an Intermediate.

Each Subordinate Statement is verifiable over time and MUST have an expiration date. The revocation of each statement is verifiable in real time and online (only for remote flows) through the federation endpoints.

Note

The revocation of an Entity is made with the unavailability of the Subordinate Statement related to it. If the Trust Anchor or its Intermediate doesn't publish a valid Subordinate Statement, or if it publishes an expired/invalid Subordinate Statement, the subject of the Subordinate Statement MUST be intended as not valid or revoked.

The concatenation of the statements, through the combination of these signing mechanisms and the binding of claims and public keys, forms the Trust Chain.

The Trust Chains can also be verified offline, using one of the Trust Anchor's public keys.

Note

Since the Wallet Instance is not a Federation Entity, the Trust Evaluation Mechanism related to it **requires the presentation of the Wallet Attestation during the credential issuance and presentation phases**.

The Wallet Attestation conveys all the required information pertaining to the instance, such as its public key and any other technical or administrative information, without any User's personal data.

Role in Onboarding: Trust evaluation mechanisms are essential during entity registration and lifecycle management to verify the compliance status of new participants and validate changes to existing entities. This includes validating the registration credentials of entities requesting onboarding and ensuring they meet federation policies before granting operational authorization.

Role in Operations: During credential issuance and presentation operations, trust evaluation provides real-time verification of entity validity and compliance status. This enables secure credential transactions by ensuring that all participating entities (credential issuers, relying parties, wallet providers) maintain their authorized status and comply with current federation policies.

6.12.1 Establishing Trust with Credential Issuers

In the issuance process, Trust Evaluation ensures the integrity and authenticity of the Credentials being issued and the reliability of their Issuers. This section delineates the Trust Evaluation mechanisms distinct from the protocol flows, implemented by Wallet Instances and Relying Parties, as described in the dedicated section.

Trust Evaluations implement different ways, as defined below:

- **Federation Entity Discovery**: Wallet Instances and Relying Parties MUST verify the identity of the Issuer using the Federation Entity Discovery process defined in *Federation Discovery* (page 125). This involves querying federation endpoints to confirm the Issuer's validity status and compliance with the Trust Framework. Historical event data from the Federation Subordinate Events Endpoint can provide additional context about the entity's lifecycle and compliance history.
- **Trust Chains**: Wallet Instances and Relying Parties evaluate Issuer's Trust Chains using the mechanisms defined in *Trust Chain* (page 127). Trust Chains may be provided statically or built through a Federation Entity Discovery process, to ensure that the entity requesting the Credential is part of a recognized and trusted federation.

- **Trust Marks Evaluation**: Trust Marks are assessed to ensure ongoing compliance with federation policies. These marks indicate adherence to specific standards and practices required by the federation.
- **Policy Evaluation**: Wallet Instances and Relying Parties MUST check that the Credential Issuer is allowed in the issuance of the Credential of their interest. Metadata, metadata policies and Trust Marks are used for the implementation of these checks (*WP_050a* (page 746)).

In the process represented in the sequence diagram below, the Wallet Instance uses the Federation API to discover and collect all the Credential Issuers enabled within the federation. The discovery process produces the Trust Chain. When the Trust Chain is provided statically within a signed request or Credential, it only REQUIRES to be refreshed when the internet connection is available, while it MUST be refreshed when the statically provided Trust Chain results as expired.

Note

As shown in the figure, the Trust Evaluation process is entirely separate and distinct from the protocol-specific flow. It operates in a different flow and utilizes specialized protocols designed specifically for this purpose.

6.12.2 Establishing Trust with Relying Party

In the context of evaluating Relying Parties, the responsibility for Trust Evaluation lies solely with the Wallet Instance. The Trust Evaluation mechanisms are distinct from protocol flows and are implemented by the Wallet Instance, as detailed in the dedicated section.

Trust Evaluations are conducted as follows:

- **Federation Entity Discovery**: When the Wallet Instance receives a signed request issued by a Relying Party, the Wallet Instance MUST verify the identity of the Relying Party using the Federation Entity Discovery process defined in *Federation Discovery* (page 125). This involves querying federation endpoints to confirm the Relying Party's validity status and compliance with the Trust Framework and evaluating the request signature using the cryptographic material obtained from the Trust Chain. Historical event data from the Federation Subordinate Events Endpoint can provide additional context about the Relying Party's lifecycle and compliance history.
- **Trust Chains**: The Wallet Instance evaluates the Relying Party's Trust Chains using the mechanisms defined in *Trust Chain* (page 127). Trust Chains may be provided statically or built through a Federation Entity Discovery process, to ensure that the Relying Party is part of a recognized and trusted federation. This involves checking the Trust Chain from the root authority (Trust Anchor) to the Relying Party (*WP_079* (page 754)).

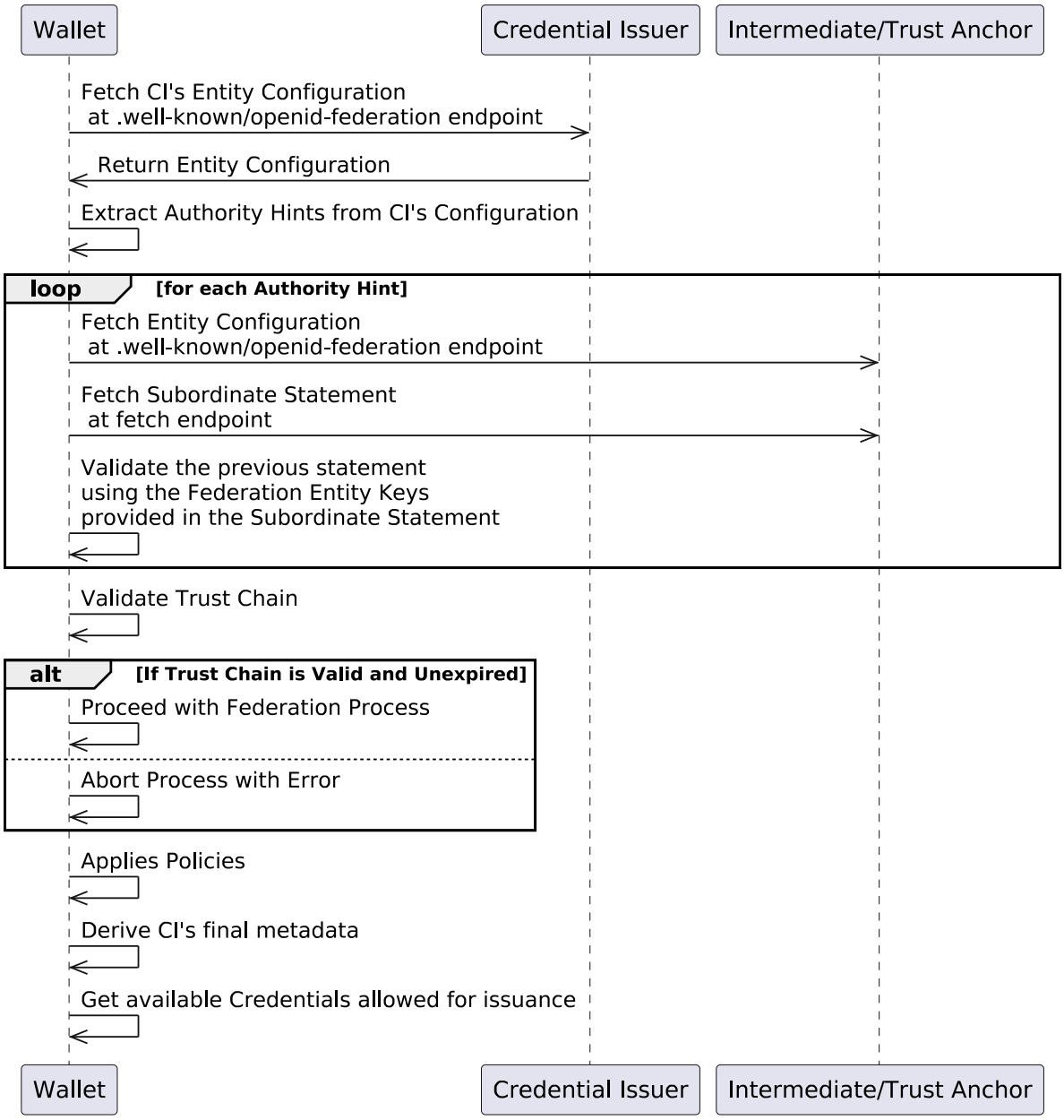


Fig. 6.2: Trust Evaluation Process.

- **Trust Marks Evaluation**: Trust Marks are assessed to ensure ongoing compliance with federation policies. These marks indicate adherence to specific standards and practices required by the federation. Relying Parties MAY include Trust Marks in their Entity Configuration to signal administrative properties and compliance to specific profiles, such as the grants in interacting with under-age users (*WP_080* (page 754)).
- **Policy Evaluation**: The Wallet Instance MUST verify that the Relying Party is authorized to request the Credential of interest. Metadata, metadata policies, and Trust Marks are used to implement these checks (*WP_087* (page 754)).

In the process depicted in the sequence diagram below, the Wallet Instance uses the Federation API to discover and collect all the Relying Parties enabled within the federation. The discovery process produces the Trust Chain. When the Trust Chain is provided statically within a signed request, it only needs to be refreshed when an internet connection is available, but it MUST be refreshed if the statically provided Trust Chain is expired.

Note

As shown in the figure, internet connection is required to update the Trust Chain about an RP and check its revocation status.

Establishing Trust with a Relying Party Operating through an Intermediary

When a Relying Party operates through a Relying Party Intermediary, the Wallet Instance MUST perform an extended Trust Evaluation process that includes validation of the Relying Party Intermediary's Trust Mark. This process takes place during the Trust Chain construction and validation phase and enables the Wallet to present to the User transparent information about the entity that registered and vouches for the RP.

The process is articulated in the following steps:

1. **Download of the RP Entity Configuration**: The Wallet Instance downloads the Relying Party's Entity Configuration from the RP's `/.well-known/openid-federation` endpoint. The RP's EC contains the `authority_hints` claim pointing to the Relying Party Intermediary's URL (not directly to the Trust Anchor).
2. **Identification of the Relying Party Intermediary**: Via the `authority_hints` claim in the RP's EC, the Wallet Instance identifies the Relying Party Intermediary and downloads its Entity Configuration.
3. **Validation of the Relying Party Intermediary's Trust Mark**: The Wallet Instance verifies that the Relying Party Intermediary's Entity Configuration contains a valid Trust Mark for the Relying Party Intermediary role, issued by the Trust Anchor. The validity of the Trust Mark MUST be verified via the Trust Anchor's `/trust_mark_status` endpoint.

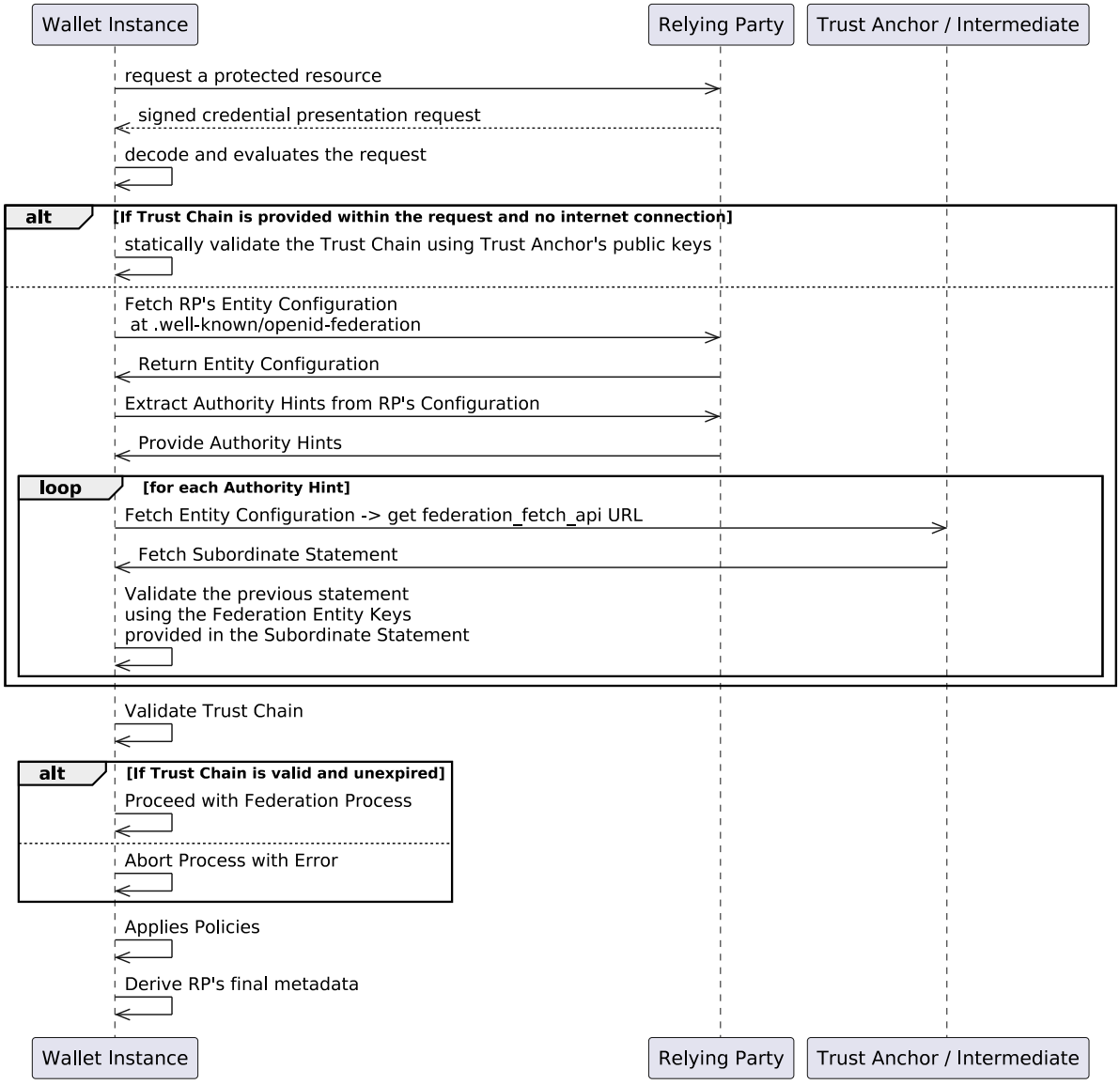


Fig. 6.3: Trust Evaluation - Relying Party Discovery Process.

4. **Trust Chain Construction through the Relying Party Intermediary**: The Wallet Instance constructs the complete Trust Chain: RP Entity Configuration → Subordinate Statement issued by the Relying Party Intermediary for the RP → Subordinate Statement issued by the Trust Anchor for the Relying Party Intermediary → Trust Anchor Entity Configuration. This chain attests that the RP is a Leaf recognized by the Relying Party Intermediary and that the Relying Party Intermediary is in turn recognized by the Trust Anchor.
5. **User Presentation**: Once validation is complete, the Wallet Instance MUST indicate to the User that the requesting Relying Party operates through a recognized Relying Party Intermediary, displaying the Relying Party Intermediary's identifying information (e.g., `organization_name`) retrieved from its Entity Configuration.

Note

The presence of the Trust Mark type ending with `federation-entity/openid_credential_verifier_intermediary` in the Relying Party Intermediary's EC is the cryptographically verifiable proof that the entity has been recognized by the Trust Anchor as an authorized Relying Party Intermediary. The Wallet Instance MUST reject the RP's request if the Trust Chain cannot be constructed and validated through a recognized Relying Party Intermediary or directly through the Trust Anchor.

6.12.3 Evaluating Trust with Wallets

The Wallet Provider issues the Wallet Attestation, certifying the operational status of its Wallet Instances and including one of their public keys.

The Wallet Attestation MAY contain the Trust Chain that attests the reliability for its issuer (Wallet Provider) at the time of issuance.

The Wallet Instance provides its Wallet Attestation within the signed request during the PID/ IT-Wallet ID issuance phase. The Credential Issuer MUST evaluate the Trust Chain about the Wallet Attestation issuer (formally, the Wallet Provider).

6.13 Non-repudiability of the Long Lived Attestations

The Trust Anchor and its Intermediate MUST expose the Federation Historical Keys endpoint, where are published all the public part of the Federation Entity Keys that are no longer used, whether expired or revoked.

The details of this endpoint are defined in the OID-FED Section 8.7.

Each JWT containing a Trust Chain in the JWT headers can be verified over time, since the entire Trust Chain is verifiable using the Trust Anchor's public key.

Even if the Trust Anchor has changed its cryptographic keys for digital signature, the Federation Historical Keys endpoint always makes the keys no longer used available for historical signature verifications.

6.14 X.509 PKI

The X.509 Public Key Infrastructure (PKI) is a framework designed to create, manage, distribute, use, store, and revoke digital X.509 Certificates. At the heart of X.509 PKI is the concept of a Certificate Authority (CA), which issues digital certificates to entities. These certificates are required for establishing secure communications over networks, including the internet, by enabling encryption and digital signature functionalities. The PKI hierarchy typically involves a root CA at the top, with one or more subordinate CAs beneath, forming a trusted chain. Entities rely on this chain of trust to verify the authenticity of certificates. X. 509 standards define the format of public key certificates.

The integration of OpenID Federation 1.0 with the traditional X.509 based PKI (rfc:5280), complemented by a RESTful API, aims to enhance the infrastructure with additional features, making it navigable and transparent.

This approach leverages the dynamic and flexible nature of OpenID Federation alongside the requirement of the X.509 Certificates for legacy applications and interoperability purposes, aiming to addresses the evolving needs of verification of the registration status of the federation participants, their compliance to the shared rules and the general and interoperable trust management in multilateral digital ecosystems.

Role in Onboarding: During entity registration, X.509 Certificates complement OpenID Federation mechanisms by providing interoperability with legacy systems and enabling integration with existing PKI infrastructures.

Role in Operations: During Credential operations, X.509 Certificates enable secure communications with legacy systems and provide alternative verification paths for entities that require traditional PKI validation.

OpenID Federation and X.509 based PKI share several things in common, as listed below:

- **Hierarchical Approach**: both utilize a hierarchical Trust Model with a single, overarching trusted third party, known as the Trust Anchor, which is trusted above all others.
- **Scalability with Multiple Trust Anchors and Intermediates**: despite a unique hierarchical model, the possibility of having multiple Trust Anchors and Intermediates, below one or more Trust Anchors, scales the responsibilities.

- **Custom Extensions**: both systems allow for custom extensions to meet specific requirements or to enhance functionality. X.509 Certificates support custom extensions, Open ID Federation allows definition of custom protocol specific metadata, Trust Marks and policies using a Policy Language.
- **Trust/Certificate Chain**: they rely on a chained proof of trust, where trust is passed down from the root authority (Trust Anchor) through Intermediaries to the end entity (Leaf).
- **Constraints in the Chain**: constraints can be applied within the Trust Chain regarding critical aspects such as the delegation of trust, the number of intermediaries, and the domains involved.
- **Public Key Distribution**: Both systems involve the distribution of the public key of the Trust Anchor to ensure entities can verify the trust chain.
- **Registry of Expired Keys**: Maintaining a registry of expired keys is crucial for both, ensuring non-repudiation of past signatures even when keys change.

6.14.1 Federation Trust Anchor and X.509 CA

In the context of OpenID Federation, the Trust Anchor plays a role similar to that of a Certificate Authority (CA) in X.509-based Public Key Infrastructures (PKIs). Both serve as foundational elements of trust within their respective systems. In this document, the term "Trust Anchor" is often used to encompass both concepts. The trust infrastructure described here aligns the Open ID Federation Trust Anchor with the X.509 PKI Certificate Authority, making therefore them a single unique entity supporting both RFC 5280 and OpenID Federation 1.0.

6.14.2 X.509 Certificates Issuance

In an OpenID Federation, each participant is required to self-issue its Entity Configuration, signing it with one of its cryptographic keys that are attested by Immediate Superiors.

In the same way, each federation Entity has the autonomy to issue a signed statement about itself in the form of a X.509 Certificate. Federation participants that need to issue X.509 Certificates about themselves and for their specific purposes, can issue and sign X.509 Certificates using one of their Federation Entity Keys attested by their Federation Authorities (Immediate Superior). This process aligns the issuance of X.509 Certificates with the federation's delegation paradigm.

This is feasible because the X.509 Certificate can be verified using a X.509 Certificate Chain, similar to the approach used for Entity Configurations in OpenID Federation.

Federation Leaves are not Certificate Authorities (CAs) or CA intermediaries authorized to issue X.509 Certificates for their subordinates. Instead, Federation Leaves act as intermediaries for issuing certificates solely about themselves. This is accomplished by applying appropriate

naming constraints to ensure that X.509 Certificates are correctly scoped. Naming constraints are applied by Immediate Superiors within the certificates issued to the Leaf entity, specifically concerning the Leaf's Federation Entity Keys. As a result, the Leaf can only issue X.509 Certificates about itself, thereby maintaining the integrity of the Trust Chain.

When a participant self-issues an X.509 Certificate, it adheres to the following requirements:

1. **Subject Name**: The X.509 Certificate's subject name MUST match the participant's identity. The subject name for Intermediaries and Leaves MUST include the following attributes:

Attribute	Requirement
`Country Name (C)`	MUST contain the two-letter ISO country code.
`State or Province Name (ST)`	MUST contain the region or state where the entity is located.
`Locality Name (L)`	MUST contain the city where the entity is located.
`Organization Name (O)`	MUST contain the legal name of the organization.
`Organizational Unit Name (OU)`	MAY contain the department name within the organization (optional).
`Common Name (CN)`	MUST contain the Federation Entity unique identifier DNS name, which is included in the `sub` (subject) value in its federation Entity Configuration, removing `https://` and any webpaths.
`Email Address`	MUST contain the organization's contact email address.
`organizationIdentifier`	MUST contain the registration number that uniquely identifies the organization within the registration service, using the OID value `2.5.4.97` as defined in `ITU-T X.500`.

2. **Subject Alternative Name (SAN)**: The X.509 Certificate MUST include a `SAN URI` that MUST match the **sub** and the **iss** values of its federation Entity Configuration.

3. **DNS Name**: The X.509 Certificate MUST include a DNS Name in the SAN that matches the DNS name contained within the **sub** and the **iss** values of its Entity Configuration, removing `https://` and any webpaths.

4. **Certificate Revocation List (CRL)**: If the issued X.509 Certificates has an expiration time superior to 24 hours, the X.509 Issuer MUST publish a CRL for the issued X.509 Certificates. This list MUST be accessible and regularly updated to ensure that any compromised or invalid X.509 Certificates are promptly revoked with the motivation of the revocation, if any.

5. **Basic Constraints**: The X.509 Certificate MUST include a `Basic Constraints` extension with `CA:TRUE` and a maximum path length of 1 if the certificate issuer is a Federation Intermediate. If it is a Leaf, the maximum path length MUST be set to 0. This indicates that the Subordinate to which certificate is about, can only issue X.509 Certificates about itself. `BasicConstraints` extension MUST be set `critical`.

6. **Key Usage**: `Digital Signature`, `Key Encipherment`, `Certificate Sign`, `CRL Sign` MUST be included. `KeyUsage` extension MUST be set `critical`.

7. **Name Constraints**: The X.509 Certificate MUST include `Name Constraints` to specify permitted and excluded domains and URIs. For instance:

   - Permitted: - `URI.1=https://leaf.example.com` - `DNS.1=leaf.example.com`
   - Excluded: - `DNS=localhost` - `DNS=localhost.localdomain` - `DNS=127.0.0.1` - `DNS=example.com` - `DNS=example.org` - `DNS=example.net` - `DNS=*.example.org`

8. **AuthorityKeyIdentifier**: The X.509 Certificate MUST include an `AuthorityKeyIdentifier` extension. The `keyIdentifier` field of the `AuthorityKeyIdentifier` extension MUST be present and MUST be identical to the `SubjectKeyIdentifier` field of the issuer's certificate. This consolidates the certificate chain building and validation.

Below is a non-normative example, in plain text (OpenSSL format), of an X.509 certificate chain with an intermediate CA, starting from the Leaf certificate.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 950713202307300703483739727303172061473531557 2
        (0x1aa5085f6cfe919f8f4ae34a8ca97e2da1f0274)
    Signature Algorithm: sha256
        Issuer: commonName=intermediate.example.org, countryName=IT,
        emailAddress=info@intermediate.example.org, organizationName=Example Intermediate
        Validity
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=leaf.example.it, countryName=IT,
        emailAddress=info@leaf.example.it, organizationName=Example Leaf
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X: 62908053918297774357647421014545143197415914813624128496938845341292839993 55
                Y:
                11297687331593767721173034223577867786983294082539048671274781317651547888 5564
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:1
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://leaf.example.it/crl/leaf.example.it.crl
            X509v3 Name Constraints:
                Permitted:
                  DNS.1=leaf.example.it
```

```
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
                  IP.3=127.0.0.1/32
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:https://leaf.example.it/openid4vp
                DNS:leaf.example.it
            X509v3 Subject Key Identifier:
                10:45:d4:c1:22:e0:f1:c2:1f:a2:84:42:df:71:0f:ba:88:16:e0:a9
            X509v3 Authority Key Identifier:
                keyid:07:4b:be:d2:bf:57:db:ce:40:e4:fa:c2:89:27:db:ff:84:8f:7c:7c

    Signature Algorithm: sha256
     30:44:02:20:64:3f:1f:a5:15:b0:62:0c:1b:c2:a3:06
     7f:e0:16:0c:cb:a4:e5:7b:70:e3:31:3a:67:5a:a9:80
     24:23:af:f1:02:20:03:fe:75:fa:75:66:1f:1c:84:30
     c3:74:57:c7:25:3b:58:05:bf:38:b9:0f:b7:88:cb:9f
     3a:6e:0c:60:61:8b
```

```
-----BEGIN CERTIFICATE-----
MIIDIzCCAsqgAwIBAgIUAapQhfbP6Rn49K40qMqX4tofAnQwCgYIKoZIzj0EAwIw
fTEhMB8GA1UEAwwYaW50ZXJtZWRpYXRlLmV4YW1wbGUub3JnMQswCQYDVQQGEwJJ
VDEsMCoGCSqGSIb3DQEJARYdaW5mb0BpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcx
HTAbBgNVBAoMFEV4YW1wbGUgSW50ZXJtZWRpYXRlMB4XDTI1MDkyODEwMTQzOFoX
DTI2MDkyOTEwMTQzOFowYzEYMBYGA1UEAwwPbGVhZi5leGFtcGxlLml0MQswCQYD
VQQGEwJJVDEjMCEGCSqGSIb3DQEJARYUaW5mb0BsZWFmLmV4YW1wbGUuaXQxFTAT
BgNVBAoMDEV4YW1wbGUgTGVhZjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABA3o
eDyFgrsWCLfLka1OAaQ5KGFyB0gZuLFMcT2GAqZ7+cak7eOb0wrX9c+n6vhlkg8D
hTD5luafdkOZUXSbFLyjggFAMIIBPDASBgNVHRMBAf8ECDAGAQH/AgEBMEAGA1Ud
HwQ5MDcwNaAzoDGGL2h0dHBzOi8vbGVhZi5leGFtcGxlLml0L2NybC9sZWFmLmV4
YW1wbGUuaXQuY3JsMFUGA1UdHgEB/wRLMEmgEzARgg9sZWFmLmV4YW1wbGUuaXSh
MjALgglsb2NhbGhvc3QwF4IVbG9jYWxob3N0LmxvY2FsZG9tYWluMAqHCH8AAAH/
////MA4GA1UdDwEB/wQEAwIBpjA9BgNVHREENjA0hiFodHRwczovL2xlYWYuZXhh
bXBsZS5pdC9vcGVuaWQ0dnCCD2xlYWYuZXhhbXBsZS5pdDAdBgNVHQ4EFgQUEEXU
wSLg8cIfooRC33EPuogW4KkwHwYDVR0jBBgwFoAUB0u+0r9X285A5PrCiSfb/4SP
fHwwCgYIKoZIzj0EAwIDRwAwRAIgZD8fpRWwYgwbwqMGf+AWDMuk5Xtw4zE6Z1qp
gCQjr/ECIAP+dfp1Zh8chDDDdFfHJTtYBb84uQ+3iMufOm4MYGGL
-----END CERTIFICATE-----
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 148177522658585329990858238295912831877447595707
        (0x19f4828a0290c333d92efe8d9c5306b979e6b6bb)
    Signature Algorithm: sha256
```

```
        Issuer: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Validity
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=intermediate.example.org, countryName=IT,
        emailAddress=info@intermediate.example.org, organizationName=Example Intermediate
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X: 98817418608085252616249916893677319857297304171504928731420155197485062951 27
                Y:
                100724824207124402297793814607603963819791163379056466342838110166095246764680
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:2
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://intermediate.example.org/crl/intermediate.example.org.crl
            X509v3 Name Constraints:
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
                  IP.3=127.0.0.1/32
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:https://intermediate.example.org
                DNS:intermediate.example.org
            X509v3 Subject Key Identifier:
                07:4b:be:d2:bf:57:db:ce:40:e4:fa:c2:89:27:db:ff:84:8f:7c:7c
            X509v3 Authority Key Identifier:
                keyid:95:b9:12:12:d0:c2:48:cc:20:2d:6c:54:96:f3:a2:7c:72:68:5c:31

    Signature Algorithm: sha256
     30:45:02:21:00:c2:98:24:30:7e:4a:69:bc:d2:b8:e0
     1c:aa:c5:98:34:57:63:d3:7f:3f:dc:4e:53:30:6e:4d
     c5:6d:64:2e:b2:02:20:2a:08:81:86:a4:06:c5:7f:84
     ab:7b:9e:1c:42:b7:17:84:c7:9c:04:a4:8a:84:22:50
     3d:71:c2:e9:10:28:54

-----BEGIN CERTIFICATE-----
MIIDJTCCAsugAwIBAgIUGfSCigKQwzPZLv6NnFMGuXnmtrswCgYIKoZIzj0EAwIw
XzEXMBUGA1UEAwwOY2EuZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZI
hvcNAQkBFhNpbmZvQGNhLmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENB
MB4XDTI1MDkyODEwMTQzOFoXDTI2MDkyOTEwMTQzOFowfTEhMB8GA1UEAwwYaW50
```

```
ZXJtZWRpYXRlLmV4YW1wbGUub3JnMQswCQYDVQQGEwJJVDEsMCoGCSqGSIb3DQEJ
ARYdaW5mb0BpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcxHTAbBgNVBAoMFEV4YW1w
bGUgSW50ZXJtZWRpYXRlMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEFdjeIOVq
+Ah2AagzWx69sHvjV2K7OkBPTyxnE63Qm1fesDruwn8LexbY0Y9qfdhpyMPaSR+A
3R988KLkypGqiKOCAUUwggFBMBIGA1UdEwEB/wQIMAYBAf8CAQIwUgYDVR0fBEsw
STBHoEWgQ4ZBaHR0cHM6Ly9pbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcvY3JsL2lu
dGVybWVkaWF0ZS5leGFtcGxlLm9yZy5jcmwwQAYDVR0eAQH/BDYwNKEyMAuCCWxv
Y2FsaG9zdDAXghVsb2NhbGhvc3QubG9jYWxkb21haW4wCocIfwAAAf////8wDgYD
VR0PAQH/BAQDAgGmMEUGA1UdEQQ+MDyGIGh0dHBzOi8vaW50ZXJtZWRpYXRlLmV4
YW1wbGUub3JnghhpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcwHQYDVR0OBBYEFAdL
vtK/V9vOQOT6wokn2/+Ej3x8MB8GA1UdIwQYMBaAFJW5EhLQwkjMIC1sVJbzonxy
aFwxMAoGCCqGSM49BAMCA0gAMEUCIQDCmCQwfkppvNK44ByqxZg0V2PTfz/cTlMw
bk3FbWQusgIgKgiBhqQGxX+Eq3ueHEK3F4THnASkioQiUD1xwukQKFQ=
-----END CERTIFICATE-----

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 660157339746673467767137473267076395071677825012
        (0x73a27a9ad0b1a3c0915c78404a7ea644a1de7ff4)
    Signature Algorithm: sha256
        Issuer: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Validity
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X:
                60510079538073841265704016785331040638654008648926352025877164647615009972064
                Y:
                94446385337085233641787598544598093044189082897506261545395904518792084782698
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://ca.example.com/crl/ca.example.com.crl
            X509v3 Name Constraints:
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
                  IP.3=127.0.0.1/32
```

```
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:https://ca.example.com
                DNS:ca.example.com
            X509v3 Subject Key Identifier:
                95:b9:12:12:d0:c2:48:cc:20:2d:6c:54:96:f3:a2:7c:72:68:5c:31

    Signature Algorithm: sha256
     30:45:02:21:00:c5:98:db:a0:5a:cd:d2:fe:a7:f3:c2
     12:b4:f5:2f:14:d2:a7:cc:17:ea:5e:ce:b2:69:7e:a9
     0f:5a:95:ab:a6:02:20:6d:b7:9b:e0:22:01:3e:e0:a0
     ac:2a:51:53:e8:73:e3:0d:b4:fb:fb:af:dc:91:56:3b
     14:c2:31:c9:d4:41:49

-----BEGIN CERTIFICATE-----
MIICuTCCAl+gAwIBAgIUc6J6mtCxo8CRXHhASn6mRKHef/QwCgYIKoZIzj0EAwIw
XzEXMBUGA1UEAwwOY2EuZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZI
hvcNAQkBFhNpbmZvQGNhLmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENB
MB4XDTI1MDkyODEwMTQzOFoXDTI2MDkyOTEwMTQzOFowXzEXMBUGA1UEAwwOY2Eu
ZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZIhvcNAQkBFhNpbmZvQGNh
LmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENBMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEhcd9yAMYDQ3pJ6mQy8nFpTOTS5khBJ35dLryF+z832DQzsJ+
Bm2AgrtqOOioLRLrSz1UNSXkY08aVGr0vcz+aqOB+DCB9TAPBgNVHRMBAf8EBTAD
AQH/MD4GA1UdHwQ3MDUwM6AxoC+GLWh0dHBzOi8vY2EuZXhhbXBsZS5jb20vY3Js
L2NhLmV4YW1wbGUuY29tLmNybDBABgNVHR4BAf8ENjA0oTIwC4IJbG9jYWxob3N0
MBeCFWxvY2FsaG9zdC5sb2NhbGRvbWFpbjAKhwh/AAAB/////zAOBgNVHQ8BAf8E
BAMCAaYwMQYDVR0RBCowKIYWaHR0cHM6Ly9jYS5leGFtcGxlLmNvbYIOY2EuZXhh
bXBsZS5jb20wHQYDVR0OBBYEFJW5EhLQwkjMIC1sVJbzonxyaFwxMAoGCCqGSM49
BAMCA0gAMEUCIQDFmNugWs3S/qfzwhK09S8U0qfMF+pezrJpfqkPWpWrpgIgbbeb
4CIBPuCgrCpRU+hz4w20+/uv3JFWOxTCMcnUQUk=
-----END CERTIFICATE-----
```

Using the underlying layer established with OpenID Federation 1.0, all X.509 Certificates are issued in a properly decentralized manner using the delegation pattern.

6.14.3 X.509 Certificate Revocation

An X.509 Certificate can be revoked by its Issuer. Revocation lists, and/or any other revocation control mechanism, are particularly required for X.509 Certificates with an expiration time greater than 24 hours; otherwise, they are not required.

When the issuer of the X.509 Certificate is the Leaf and therefore the X.509 Certificate refers to itself, if the certificate expiration time is greater than 24 hours from the `X509_NOT_VALID_BEFORE` time, it MUST implement a CRL related to the issued certificate and keep it updated. When the issuer of the X.509 Certificate is an immediate Superior, such as the Trust Anchor

or an Intermediary, and revokes the certificate related to the Leaf, i.e., the X.509 Certificate related to one of the Federation Entity Keys of the Leaf, this action invalidates the entire Trust Chain associated with that cryptographic public key of the Leaf, effectively removing its ability to issue further X.509 Certificates about itself. This hierarchical revocation mechanism ensures that any compromise or misbehavior by a Leaf entity can be quickly addressed.

Below is a non-normative example, in plain text, illustrating the content of a CRL.

```
Certificate Revocation List (CRL):
Version: 2 (0x1)
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=leaf.example.org, O=Leaf, C=IT
Last Update: Sep 1 00:00:00 2023 GMT
Next Update: Sep 8 00:00:00 2023 GMT
Revoked Certificates:
    Serial Number: 987654320
        Revocation Date: Aug 25 12:00:00 2023 GMT
        CRL Entry Extensions:
            Reason Code: Key Compromise
    Serial Number: 987654321
        Revocation Date: Aug 30 15:00:00 2023 GMT
        CRL Entry Extensions:
            Reason Code: Cessation of Operation
Signature Algorithm: sha256WithRSAEncryption
Signature:
    5c:4f:3b:...
```

6.14.4 Federation Subordinate Events Endpoint

The Federation Subordinate Events Endpoint is defined in OID-FED-SUBORDINATE-EVENTS. This endpoint provides a mechanism for Trust Anchors and Intermediates to publish historical events related to their Immediate Subordinates registration status. It provides transparency and accountability within the federation by providing a comprehensive historical record of significant events affecting federation participants.

For complete specification details, including endpoint location, request format, response format, JWT claims, event object parameters, and supported event types, refer to OID-FED-SUBORDINATE-EVENTS.

Below is a non-normative example of a Federation Subordinate Events Endpoint request and response:

Example Request:

```
GET /federation_subordinate_events_endpoint?sub=https%3A%2F%2Frp%2Eexample%2Eorg HTTP/1.1
Host: immediate-superior.example.org
```

Example Response:


```
{
  "iss": "https://immediate-superior.example.org",
  "sub": "https://rp.example.org",
  "iat": 1590000000,
  "federation_registration_events": [
    {
      "iat": 1590000000,
      "event": "registration"
    },
    {
      "iat": 1590000000,
      "event": "jwks_update"
    },
    {
      "iat": 1600000000,
      "event": "revocation",
      "event_description": "compromised node"
    },
    {
      "iat": 1610000000,
      "event": "registration"
    }
  ]
}
```


Integration with Entity Lifecycle Management:

This endpoint complements the entity lifecycle management procedures defined in *Entity Onboarding* (page 225) by providing detailed historical tracking of all significant events affecting federation participants. It supports both automated compliance monitoring and manual auditing processes.

6.15 Privacy Remarks

- Wallet Instances MUST NOT publish their metadata through an online service.
- The trust infrastructure MUST be public, with all endpoints publicly accessible without any client credentials that may disclose who is requesting access.
- When a Wallet Instance requests the Subordinate Statements to build the Trust Chain for a specific Relying Party or validates a Trust Mark online, issued for a specific Relying Party, the Trust Anchor or its Intermediate do not know that a particular Wallet Instance is inquiring about a specific Relying Party; instead, they only serve the statements related to that Relying Party as a public resource.

- The Wallet Instance metadata MUST NOT contain information that may disclose technical information about the hardware used.
- Leaf entity, Intermediate, and Trust Anchor metadata may include the necessary amount of data as part of administrative, technical, and security contact information. It is generally not recommended to use personal contact details in such cases. From a legal perspective, the publication of such information is needed for operational support concerning technical and security matters and the GDPR regulation.

6.16 Considerations about Decentralization

- There may be more than a single Trust Anchor.
- In some cases, a trust verifier may trust an Intermediate, especially when the Intermediate acts as a Trust Anchor within a specific perimeter, such as cases where the Leafs are both in the same perimeter like a Member State jurisdiction (eg: an Italian Relying Party with an Italian Wallet Instance may consider the Italian Intermediate as a Trust Anchor for the scopes of their interactions).
- Trust attestations (Trust Chain) should be included in the JWT issued by Credential Issuers, and the Presentation Requests of RPs should contain the Trust Chain related to them (issuers of the presentation requests).
- Since the credential presentation must be signed, storing the signed presentation requests and responses, which include the Trust Chain, the Wallet Instance may have the snapshot of the federation configuration (Trust Anchor Entity Configuration in the Trust Chain) and the verifiable reliability of the Relying Party it has interacted with.
- Each signed attestation is long-lived since it can be cryptographically validated even when the federation configuration changes or the keys of its issuers are renewed.
- Each participant should be able to update its Entity Configuration without notifying the changes to any third party. The metadata policy contained within a Trust Chain must be applied to overload any information related to protocol specific metadata.

CHAPTER

SEVEN

REGISTRY INFRASTRUCTURE

The IT-Wallet ecosystem operates through a registry infrastructure that provides standardized data definitions, entity registration, and Credential discovery capabilities. The registry system consists of multiple interconnected components that support the complete lifecycle of digital Credential operations from entity onboarding to Credential presentation.

The registry architecture addresses semantic standardization, federation trust management, and Credential discovery requirements through specialized registry components that ensure interoperability and compliance across the ecosystem.

7.1 Registry Architecture Overview

The IT-Wallet System Register comprises six main components:

1. **Claims Registry**: Standardized semantic definitions for individual Credential attributes, data types, and validation rules.
2. **Authentic Source (AS) Registry**: Catalog of registered data providers with their declared capabilities and available claims.
3. **Federation Registry**: Authoritative list of trusted entities participating in the federation with their technical configurations.
4. **Digital Credentials Catalog**: Public discovery mechanism for available Credential types with their metadata and issuance information.
5. **Schema Registry**: Authoritative list of Credential Schemas.
6. **Taxonomy**: Hierarchical classification system organizing Credentials by domain and purpose.

These registry components are interconnected and maintained by the Supervisory Body to ensure consistency, security, and regulatory compliance across the ecosystem.

7.2 Registry Discovery Endpoint

The Trust Anchor MUST provide a discovery mechanism for all registry components through standardized *well-known* endpoints providing metadata and REST API discovery information to handle complex operations like pagination and filtering.

The Trust Anchor MUST publish registry discovery metadata at the `.well-known/it-wallet-registry` endpoint with content negotiation support:

- **Default Content-Type**: `application/jwt` (signed JWT ensuring authenticity and integrity)
- **Alternative Content-Type**: `application/json` (plain JSON for development/debugging purposes)

Moreover, the IT-Wallet System Register MUST use two distinct access patterns:

- **Data Registry APIs**: MUST support pagination and filtering capabilities.
- **Federation Trust Infrastructure**: as defined in *The Infrastructure of Trust* (page 109).

Below a non-normative example is given.

```
GET /.well-known/it-wallet-registry HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/jwt

HTTP/1.1 200 OK
Content-Type: application/jwt

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

```
GET /.well-known/it-wallet-registry HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
```

7.2.1 Registry Discovery Endpoint Parameters

The JWT payload of the Registry Discovery response MUST contain the following parameters:

Table 7.1: Registry Discovery Endpoint — JWT Payload Parameters

Field Name	Description
id	REQUIRED. Unique identifier of the discovery document (e.g., `urn:it-wallet-registry:it-wallet`).
version	REQUIRED. Version of the discovery document format (e.g., `1.0.0`).
last_modified	REQUIRED. Timestamp of the last modification to the discovery document (e.g., `2024-03-15T10:30:00Z`).
endpoints	REQUIRED. JSON object containing the URIs of all registry components. The following endpoint keys MUST be present: • **claims_registry**: URI of the Claims Registry API. • **authentic_sources**: URI of the Authentic Source Registry API. • **credential_catalog**: URI of the Digital Credentials Catalog well-known endpoint. • **taxonomy**: URI of the Taxonomy resource. • **schema_registry**: URI of the Schema Registry API. • **federation_list**: URI of the federation list endpoint (OpenID Federation `/list`). • **federation_fetch**: URI of the federation fetch endpoint (Open ID Federation `/fetch`). • **federation_resolve**: URI of the federation resolve endpoint (OpenID Federation `/resolve`). • **federation_trust_mark_status**: URI of the Trust Mark status endpoint. • **federation_historical_keys**: URI of the historical JWKs endpoint.
content_negotiation	REQUIRED. Array of content types supported by the discovery endpoint (e.g., `["application/json", "application/jwt"]`).

JWT payload structure (when decoded):


```
{
  "id": "urn:it-wallet-registry:it-wallet",
  "version": "1.0.0",
  "last_modified": "2024-03-15T10:30:00Z",
  "endpoints": {
    "claims_registry": "https://trust-anchor.eid-wallet.example.it/api/v1/claims",
    "authentic_sources": "https://trust-anchor.eid-wallet.example.it/api/v1/authentic-sources",
    "credential_catalog": "https://trust-anchor.eid-wallet.example.it/api/v1/.well-known/credential-catalog",
    "taxonomy": "https://trust-anchor.eid-wallet.example.it/api/v1/taxonomy",
```

```
    "schema_registry": "https://trust-anchor.eid-wallet.example.it/api/v1/schemas",
    "federation_list_endpoint": "https://trust-anchor.eid-wallet.example.it/list",
    "federation_fetch_endpoint": "https://trust-anchor.eid-wallet.example.it/federation_fetch_endpoint",
    "federation_resolve_endpoint": "https://trust-anchor.eid-wallet.example.it/resolve",
    "federation_trust_mark_status_endpoint": "https://trust-anchor.eid-wallet.example.it/trust_mark_status",
    "federation_trust_mark_list_endpoint": "https://dev.ta.wallet.ipzs.it/trust_mark_listing",
    "federation_trust_mark_endpoint": "https://dev.ta.wallet.ipzs.it/trust_mark",
    "federation_historical_keys_endpoint": "https://trust-anchor.eid-wallet.example.it/federation_historical_keys"
  },
  "content_negotiation": ["application/json", "application/jwt"]
}
```


7.3 Claims Registry

The **Claims Registry** provides standardized semantic definitions for individual Credential attributes, data types, and validation rules. This registry serves as the semantic foundation for Credential attribute standardization across the IT-Wallet ecosystem, working in coordination with the Taxonomy component for hierarchical classification.

The Supervisory Body MUST maintain the Claims Registry to ensure semantic consistency and regulatory compliance across the ecosystem. The registry MUST contain:

- **Standardised Claims**: Semantic definitions for all Credential attributes with data types and validation rules.
- **Interoperability Mappings**: Alias definitions for claims that use different terminology across standards (e.g., ISO18013-5 `place_of_birth` mapped to canonical `birth_place`).
- **Data Formats**: Standardised data types (string, date, numeric, boolean, email, url, image, array, object) with validation patterns.

The Claims Registry MUST ensure:

- **Semantic Consistency**: Prevents conflicts between duplicate or overlapping claims across the ecosystem.
- **Cross-border Interoperability**: Ensures EU compliance and consistent claim interpretation.
- **Schema Validation**: Provides authoritative definitions for claim validation across all Credential scenarios.
- **Regulatory Alignment**: Coordinates with national and EU regulatory framework.

- **Credential-Agnostic Scenarios**: Supports scenarios where **user convenience** and **business operational efficiency** are prioritized over **regulatory compliance** and **audit trails**.

Note

The Claims Registry defines semantic properties of individual attributes, but MUST NOT specify selective disclosure capabilities. Selective disclosure depends on Credential format implementations (SD-JWT, mDocs), issuer technical configurations, and presentation context. These capabilities are specified at the Credential type level within the Digital Credentials Catalog and implemented during Credential presentation flows.

7.3.1 Claims Registry Usage

The Claims Registry MUST support the complete ecosystem lifecycle:

During Onboarding Process:

- **AS Registration**: Authentic Sources declare available claims from standardized registry during capability registration.
- **CI Registration**: Credential Issuers select AS entities based on required claims and register Credential types for catalog publication.
- **RP Registration**: Relying Parties specify authorization requirements using domains/purposes for specific Credential types and/or User's attributes.

During Operational Activities:

- **Credential Issuance**: Claims definitions ensure consistent data representation across different Credential types.
- **Presentation Requests**: RPs reference claims for schema validation and authorization verification in both credential-specific and credential-agnostic scenarios.
- **Policy Enforcement**: Authorization policies leverage domain/purpose classifications for access control.

7.3.2 Claims Registry Structure

The Claims Registry maintains language-neutral, technical definitions for semantic consistency across the ecosystem. User-facing localizations for claim names and descriptions are provided via dedicated localization bundles referenced through the `localization.base_uri` field, enabling efficient multilingual support without compromising the registry's structural integrity.

Table 7.2: First-level Fields of the Claims Registry

Field Name	Description
id	REQUIRED. Unique identifier of the Claims Registry (e.g., `urn:claims:it-wallet`).
version	REQUIRED. The version of the Claims Registry (e.g., `1.0.0`).
last_modified	REQUIRED. The timestamp indicating when the Claims Registry was last updated (e.g., `2026-03-06T00:00:00Z`).
localization	REQUIRED. Localization configuration object containing: • **default_locale**: Default locale code (e.g., `it`). • **available_locales**: Array of supported locale codes (e.g., `["en", "it"]`). • **base_uri**: Base URI for localization bundle retrieval (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/`). • **version**: Version of the localization bundle format.
claims	REQUIRED. A JSON Object where each key is a claim name and each value is a JSON Object describing that claim. Each claim object contains the parameters defined in the "Claim Entry Parameters" table below.

Table 7.3: Claim Entry Parameters

Field Name	Description
description_l10n_id	REQUIRED. Localization key referencing the human-readable description of the claim in the localization bundle (e.g., `claim.given_name.description`).
type	REQUIRED. Data type of the claim. Supported values: `string`, `boolean`, `array`, `object`.
format	OPTIONAL. Semantic format qualifier for string types (e.g., `date` for ISO 8601 dates, `uri`, `data` for Base64-encoded binary).
encoding	OPTIONAL. Encoding applied to the value (e.g., `base64`). Present when `format` is `data`.

continues on next page

Table 7.3 – continued from previous page

Field Name	Description
aliases	OPTIONAL. Array of alternative claim names used in other standards that map to this canonical claim (e.g., `["birthdate"]` for `birth_date`, `["date_of_expiry"]` for `expiry_date`).
nested_claims	OPTIONAL. Array of claim names that form the properties of an `object` type claim (e.g., `["country", "locality", "region"]` for `place_of_birth`).
nested_item_claims	OPTIONAL. Array of claim names representing the properties of each item in an `array` type claim (e.g., `["vehicle_category_code", "issue_date", "expiry_date", "codes"]` for `driving_privileges`).
items	OPTIONAL. JSON object describing the schema of each element in a simple `array` type claim (e.g., `{"type": "string"}` for `nationalities`).

A non-normative example of Claims Registry structure is given below:


```
{
    "id": "urn:claims:it-wallet",
    "version": "1.0.0",
    "last_modified": "2026-03-06T00:00:00Z",
    "localization": {
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/",
        "default_locale": "it",
        "version": "1.0.0"
    },
    "claims": {
        "academic_qualification_date": {
            "description_l10n_id": "claim.academic_qualification_date.description",
            "format": "date",
            "type": "string"
        },
        "academic_year": {
            "description_l10n_id": "claim.academic_year.description",
            "type": "string"
        },
        "address": {
            "description_l10n_id": "claim.address.description",
            "nested_claims": [
                "country",
                "house_number",
                "locality",
                "locality_fraction",
```

```
            "postal_code",
            "region",
            "street_address"
        ],
        "type": "object"
    },
    "birth_date": {
        "aliases": [
            "birthdate"
        ],
        "description_l10n_id": "claim.birth_date.description",
        "format": "date",
        "type": "string"
    },
    "birth_place": {
        "description_l10n_id": "claim.birth_place.description",
        "type": "string"
    },
    "codes": {
        "description_l10n_id": "claim.codes.description",
        "items": {
            "properties": {
                "code": {
                    "description_l10n_id": "claim.code.description",
                    "type": "string"
                },
                "sign": {
                    "description_l10n_id": "claim.sign.description",
                    "type": "string"
                },
                "value": {
                    "description_l10n_id": "claim.value.description",
                    "type": "string"
                }
            },
            "type": "object"
        },
        "nested_item_claims": [
            "code",
            "sign",
            "value"
        ],
        "type": "array"
    },
    "constant_attendance_allowance": {
        "description_l10n_id": "claim.constant_attendance_allowance.description",
        "type": "boolean"
```

```
},
"consulate_code": {
    "description_l10n_id": "claim.consulate_code.description",
    "type": "string"
},
"consulate_name": {
    "description_l10n_id": "claim.consulate_name.description",
    "type": "string"
},
"content": {
    "description_l10n_id": "claim.content.description",
    "encoding": "base64",
    "format": "data",
    "type": "string"
},
"country": {
    "description_l10n_id": "claim.country.description",
    "type": "string"
},
"cpia_school_year": {
    "description_l10n_id": "claim.cpia_school_year.description",
    "type": "string"
},
"degree_class": {
    "description_l10n_id": "claim.degree_class.description",
    "type": "string"
},
"degree_class_name": {
    "description_l10n_id": "claim.degree_class_name.description",
    "type": "string"
},
"degree_course_name": {
    "description_l10n_id": "claim.degree_course_name.description",
    "type": "string"
},
"degree_course_year": {
    "description_l10n_id": "claim.degree_course_year.description",
    "type": "string"
},
"document_iss_authority": {
    "description_l10n_id": "claim.document_iss_authority.description",
    "type": "string"
},
"document_iss_country": {
    "description_l10n_id": "claim.document_iss_country.description",
    "type": "string"
},
```

```
"document_number": {
    "description_l10n_id": "claim.document_number.description",
    "type": "string"
},
"driving_privileges": {
    "description_l10n_id": "claim.driving_privileges.description",
    "nested_item_claims": [
        "vehicle_category_code",
        "issue_date",
        "expiry_date",
        "codes"
    ],
    "type": "array"
},
"education_degrees": {
    "description_l10n_id": "claim.education_degrees.description",
    "nested_item_claims": [
        "academic_qualification_date",
        "degree_class",
        "degree_class_name",
        "degree_course_name",
        "institute_name",
        "programme_type_name",
        "qualification_grade_value",
        "qualification_name"
    ],
    "type": "array"
},
"education_enrollments": {
    "description_l10n_id": "claim.education_enrollments.description",
    "nested_item_claims": [
        "academic_year",
        "degree_class",
        "degree_class_name",
        "degree_course_name",
        "degree_course_year",
        "institute_name",
        "programme_type_name"
    ],
    "type": "array"
},
"educational_track": {
    "description_l10n_id": "claim.educational_track.description",
    "type": "string"
},
"effective_date": {
    "description_l10n_id": "claim.effective_date.description",
```

```
        "format": "date",
        "type": "string"
    },
    "expiry_date": {
        "aliases": [
            "date_of_expiry"
        ],
        "description_l10n_id": "claim.expiry_date.description",
        "format": "date",
        "type": "string"
    },
    "family_name": {
        "description_l10n_id": "claim.family_name.description",
        "type": "string"
    },
    "foreign_residence": {
        "description_l10n_id": "claim.foreign_residence.description",
        "type": "boolean"
    },
    "given_name": {
        "description_l10n_id": "claim.given_name.description",
        "type": "string"
    },
    "house_number": {
        "description_l10n_id": "claim.house_number.description",
        "type": "string"
    },
    "institute_code": {
        "description_l10n_id": "claim.institute_code.description",
        "type": "string"
    },
    "institute_name": {
        "description_l10n_id": "claim.institute_name.description",
        "type": "string"
    },
    "issue_date": {
        "description_l10n_id": "claim.issue_date.description",
        "format": "date",
        "type": "string"
    },
    "link_qr_code": {
        "description_l10n_id": "claim.link_qr_code.description",
        "format": "uri",
        "type": "string"
    },
    "locality": {
        "description_l10n_id": "claim.locality.description",
```

```
        "type": "string"
    },
    "locality_fraction": {
        "description_l10n_id": "claim.locality_fraction.description",
        "type": "string"
    },
    "nationalities": {
        "description_l10n_id": "claim.nationalities.description",
        "items": {
            "type": "string"
        },
        "type": "array"
    },
    "place_of_birth": {
        "description_l10n_id": "claim.place_of_birth.description",
        "nested_claims": [
            "country",
            "locality",
            "region"
        ],
        "type": "object"
    },
    "portrait": {
        "description_l10n_id": "claim.portrait.description",
        "encoding": "base64",
        "format": "data",
        "type": "string"
    },
    "postal_code": {
        "description_l10n_id": "claim.postal_code.description",
        "type": "string"
    },
    "professional_qualification": {
        "description_l10n_id": "claim.professional_qualification.description",
        "type": "string"
    },
    "programme_type_name": {
        "description_l10n_id": "claim.programme_type_name.description",
        "type": "string"
    },
    "qualification_grade_value": {
        "description_l10n_id": "claim.qualification_grade_value.description",
        "type": "string"
    },
    "qualification_name": {
        "description_l10n_id": "claim.qualification_name.description",
        "type": "string"
```

```
},
"qualification_type": {
    "description_l10n_id": "claim.qualification_type.description",
    "type": "string"
},
"qualification_type_description": {
    "description_l10n_id": "claim.qualification_type_description.description",
    "type": "string"
},
"qualification_year": {
    "description_l10n_id": "claim.qualification_year.description",
    "type": "string"
},
"region": {
    "description_l10n_id": "claim.region.description",
    "type": "string"
},
"restrictions_conditions": {
    "description_l10n_id": "claim.restrictions_conditions.description",
    "type": "string"
},
"school_attendance_type": {
    "description_l10n_id": "claim.school_attendance_type.description",
    "type": "string"
},
"school_code": {
    "description_l10n_id": "claim.school_code.description",
    "type": "string"
},
"school_course_year": {
    "description_l10n_id": "claim.school_course_year.description",
    "type": "string"
},
"school_name": {
    "description_l10n_id": "claim.school_name.description",
    "type": "string"
},
"school_qualifications": {
    "description_l10n_id": "claim.school_qualifications.description",
    "nested_item_claims": [
        "educational_track",
        "institute_code",
        "institute_name",
        "professional_qualification",
        "qualification_grade_value",
        "qualification_type",
        "qualification_type_description",
```

```
                "qualification_year",
                "school_code",
                "school_name",
                "school_year"
            ],
            "type": "array"
        },
        "school_year": {
            "description_l10n_id": "claim.school_year.description",
            "type": "string"
        },
        "street_address": {
            "description_l10n_id": "claim.street_address.description",
            "type": "string"
        },
        "tax_id_code": {
            "description_l10n_id": "claim.tax_id_code.description",
            "type": "string"
        },
        "un_distinguishing_sign": {
            "description_l10n_id": "claim.un_distinguishing_sign.description",
            "type": "string"
        },
        "vehicle_category_code": {
            "description_l10n_id": "claim.vehicle_category_code.description",
            "type": "string"
        }
    }
}
```

Note

For a better and more efficient management of the localization of the information contained in the Claims Registry, an Entity consulting it SHOULD:

- Download the basic version of the Claims Registry (compact, without localizations) using the `.well-known/claims` endpoint.
- Determine the User's preferred language.
- Download only the necessary localization bundles.
- Dynamically merge localised content with the Claims Registry structure.

A non-normative example of a localization bundle output is given below:

```
{
  "claim.given_name.description": "Person's given name(s) as they appear on official documents.
",
  "claim.birth_date.description": "Date of birth, in ISO 8601 format (YYYY-MM-DD). Also known a
s birthdate.",
  "claim.driving_privileges.description": "Array of authorized vehicle categories with details.
",
  "...": "..."
}
```

Localization bundles MUST be available at the URI composed by appending the locale code and `.json` to the `localization.base_uri` value (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/it.json`).

7.4 Authentic Source Registry

The Supervisory Body MUST maintain the Authentic Source Registry to enable coordinated data access and Credential issuance across the ecosystem. The AS Registry MUST contain at least:

- **Organization Information**: Legal entity details, regulatory status, and authoritative role within specific domains.
- **Data Capabilities**: Declared claims availability referencing standardized definitions from the Claims Registry with corresponding Taxonomy classifications.
- **Integration Methods**: Technical access mechanisms (PDND).
- **Intended Purposes**: Supported Credential types and business contexts for AS-CI coordination.
- **Data Quality Assurance**: Authoritative status, update frequency, and audit trail capabilities.

The AS Registry MUST ensure:

- **Coordinated Data Access**: Enables CI discovery of appropriate data from Authentic Sources for Credential issuance.
- **AS-CI Integration**: Facilitates approval workflows and data access coordination between entities.
- **Quality Assurance**: Maintains authoritative status and data reliability across different domains.
- **Regulatory Compliance**: Supports public administration transparency and private sector coordination requirements.

Note

The Authentic Source Registry is a public registry that provides the Credential Issuer with guidance for Credential provisioning.

7.4.1 Authentic Source Registry Usage

The AS Registry supports ecosystem coordination throughout the operational lifecycle:

During Onboarding Process:

- **AS Self-Declaration**: Authentic Sources register capabilities before any Credential types exist in the catalog.
- **CI Discovery**: Credential Issuers search for AS entities based on required claims and intended Credential types.
- **Approval Coordination**: AS entities evaluate and approve CI access requests for data provision.

During Operational Activities:

- **Data Source Resolution**: CI systems reference AS Registry for real-time data access during Credential issuance.
- **Quality Validation**: AS Registry information supports data origin verification and audit requirements.
- **Integration Management**: Technical endpoints and access methods enable standardized AS-CI communication.

7.4.2 Public vs Private AS Coordination

The AS Registry architecture supports different coordination patterns reflecting distinct operational requirements:

1. **Public Administration AS** (Standardized Integration): Government entities provide authoritative data through regulated mechanisms:

- **PDND Integration**: `"integration_method": "pdnd"` for standardized government data access.
- **Regulatory Compliance**: Full transparency requirements with public catalog publication.
- **Audit Requirements**: Complete traceability for government Credential issuance processes.

2. **Private Sector AS** (Flexible Integration): Private entities provide specialized data through custom arrangements:

- **Custom APIs**: `"integration_method": "pdnd"` for business-specific data access.
- **Regulatory Compliance**: Full transparency requirements with public catalog publication.
- **Selective Disclosure**: Limited public visibility with CI-specific approval workflows.
- **Business Flexibility**: Tailored integration supporting diverse private sector use cases.

This approach enables both **regulatory transparency** for public administration and **business flexibility** for private sector entities while maintaining coordinated data access across the ecosystem.

7.4.3 AS Registry Structure

During registration, Authentic Sources declare their capabilities before Credential types exist in the catalog. This declaration establishes the foundation for subsequent CI registration and Credential type creation.

AS Unique Identifier Schema

Each Authentic Source MUST be assigned a unique identifier that follows the HTTPS URL schema defined below. This identifier is used for referencing AS entities across the registry system and in the Digital Credentials Catalog, ensuring consistency with OpenID Federation entity identification patterns.

AS Identifier Schema:

```
https://{organization_domain}[/{optional_path}]
```

Schema Components:

- **organization_domain**: DNS domain controlled by the organization
- **optional_path**: Additional path component for specific services or departments

The AS identifier MUST follow these normative rules:

1. **HTTPS Protocol**: MUST use HTTPS scheme for security and trust verification
2. **Domain Ownership**: Organization MUST control the DNS domain used in the identifier
3. **Uniqueness**: Guaranteed through DNS namespace uniqueness

4. **Stability**: SHOULD remain stable over time to avoid reference breakage
5. **Resolvability**: The URL SHOULD be resolvable (though not required to serve content)

Examples of compliant AS identifiers:

- `https://motorizzazione.gov.example`: Public - Ministry of Transport, Motorization Dept
- `https://registry.anpr.example`: Public - National Registry of Resident Population
- `https://api.bank.example/auth-source`: Private - Example Bank Financial Services

Authentic Source Registry Parameters

The Authentic Source Registry MUST contain the following parameters for each registered Authentic Source:

Table 7.4: First-level Fields of the Authentic Source Registry

Field Name	Description
id	REQUIRED. Unique identifier of the Authentic Source Registry (e.g., `urn:authentic-sources:it-wallet`).
version	REQUIRED. The version of the Authentic Source Registry (e.g., `1.0.0`).
last_modified	REQUIRED. The timestamp indicating when the Authentic Source Registry was last updated (e.g., `2025-03-15T12:00:00Z`).
localization	REQUIRED. Localization configuration object containing: • **default_locale**: Default locale code (e.g., `it`). • **available_locales**: Array of supported locale codes (e.g., `["en", "it"]`). • **base_uri**: Base URI for localization bundle retrieval (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/`). • **version**: Version of the localization bundle format.
authentic_sources	REQUIRED. A JSON Array where each entry is a JSON Object representing an Authentic Source entity. Each object contains the parameters defined in the "Authentic Sources Parameters" table below, including entity identification, organizational information, data capabilities, and integration methods.

Table 7.5: Authentic Sources Parameters

Parameter	Type	Description
entity_id	string	REQUIRED. Unique identifier following the normative schema: `https://{organization_domain}[/{optional_path}]`.
organization_info	JSON object	REQUIRED. Legal entity details and organizational metadata.
organization_info.organization_name_l10n_id	string	REQUIRED. Localization key referencing the localized organization name in the localization bundle (e.g., `authentic_source1.name`).
organization_info.organization_type	string	REQUIRED. Entity classification: `"public"` or `"private"`.
organization_info.ipa_code	string	REQUIRED only for Public AS. IPA registration code for government entities.
organization_info.legal_identifier	string	REQUIRED. Legal registration identifier (Fiscal Code/VAT Number, or equivalent national identifier for foreign entities).
organization_info.homepage_uri	string	REQUIRED. URL pointing to the organization's homepage.
organization_info.contacts	String Array	REQUIRED. Array of contact email addresses for at least one user-support, one application, and one systems specialist.
organization_info.dpa_contact	string	REQUIRED. An e-mail address of Authentic Source DPA.
organization_info.policy_uri	string	REQUIRED. URL to privacy policy document.
organization_info.tos_uri	string	OPTIONAL. URL to terms of service document.
organization_info.organization_country	string	REQUIRED. Two-letter ISO 3166-1 alpha-2 country code of the organization.
organization_info.logo_uri	string	OPTIONAL. URL to the organization's logo image.
organization_info.logo_uri#integrity	string	CONDITIONAL. Cryptographic digest of the logo image resource for integrity verification. REQUIRED if `logo_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).
organization_info.logo_alt_text_l10n_id	string	OPTIONAL. Alternative text for the organization's logo image.
organization_info.logo_extended_uri	string	OPTIONAL. URL to the organization's extended logo image.
organization_info.logo_extended_uri#integrity	string	CONDITIONAL. Cryptographic digest of the extended logo image resource for integrity verification. REQUIRED if `logo_extended_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).

continues on next page

Table 7.5 – continued from previous page

Parameter	Type	Description
organization_info.logo_extended_alt_text_l10n_id	string	OPTIONAL. Alternative text for the organization's extended logo image.
data_capabilities	JSON Objects Array	REQUIRED. Array containing data capability specifications.
data_capabilities[].dataset_id	string	REQUIRED. The *Dataset_id* within the scope of the Authentic Source, which MAY be used as a query parameter for the `GetAttributeClaims` service.
data_capabilities[].data_origin_l10n_id	string	REQUIRED. Localization key referencing the human-readable name of the data origin or department providing the data (e.g., `authentic_source1.dataset1.origin`).
data_capabilities[].intended_purposes	String Array	REQUIRED. Business purposes served, using taxonomy purpose identifiers (e.g., `["IDENTITY_VERIFICATION", "DRIVING_RIGHTS_VERIFICATION"]`).
data_capabilities[].available_claims	String Array	REQUIRED. Claims available from this data capability.
data_capabilities[].available_claims.claim_name	string	REQUIRED. It Contains the name of the claim.
data_capabilities[].available_claims.order	number	REQUIRED. Defines the order in which the information would be shown.
data_capabilities[].available_claims.mandatory	boolean	REQUIRED. Defines if a claim is always available or not.
data_capabilities[].integration_method	string	REQUIRED. Authorization framework used for data access. MUST be `"pdnd"`.
data_capabilities[].integration_endpoint	string	OPTIONAL. Service access point (PDND endpoint).
data_capabilities[].api_specification	string	OPTIONAL. URL to OAS3 specification document for this data capability.
data_capabilities[].data_provision	JSON object	OPTIONAL. Data provision capabilities and timing specifications.
data_capabilities[].data_provision.immediate_flow	boolean	REQUIRED. Indicates if the Authentic Source supports immediate data provision.
data_capabilities[].data_provision.deferred_flow	boolean	REQUIRED. Indicates if the Authentic Source supports deferred data provision.

continues on next page

Table 7.5 – continued from previous page

Parameter	Type	Description
data__capabilities[]. data__provision. max__response__ time__minutes	integer	CONDITIONAL. Maximum time in minutes for the Authentic Source to respond to a deferred data provision request. REQUIRED if `deferred_flow` is `true`.
data__capabilities[]. data__provision.notification__methods	String Array	CONDITIONAL. Array of notification methods supported by the Authentic Source for deferred data provision, such as `"push"`, `"poll"`. REQUIRED if `deferred_flow` is `true`.
data__capabilities[]. user__information__ l10n__id	string	OPTIONAL. Localization key referencing a Markdown-formatted string with human-readable information about the data capability relevant to the User (e.g., `authentic_source1.dataset1.userinfo`). This string MUST be provided by the Authentic Source to the Trust Anchor during onboarding. The Markdown formatting can be plain text or a combination of text and links. For example, if the Authentic Source's database only contains data registered *after* a specific date, this information MUST be conveyed through this key.
data__capabilities[]. service__documentation__uri	string	OPTIONAL. URL pointing to the Authentic Source service documentation.
data__capabilities[]. update__frequency	string	OPTIONAL. Indicates how frequently the Authentic Source updates its data. Possible values: `"real_time"` (near real-time updates, typically within minutes), `"daily"`, `"weekly"`, `"monthly"`, `"on_demand"`.
data__capabilities[]. logo__uri	string	OPTIONAL. URL to the logo image related to the data.
data__capabilities[]. logo__uri#integrity	string	CONDITIONAL. Cryptographic digest of the logo image resource for integrity verification. REQUIRED if `logo_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).
data__capabilities[]. logo__alt__text__ l10n__id	string	OPTIONAL. Alternative text for the organization's logo image.
data__capabilities[]. background__color	string	OPTIONAL. String value of the background color related to be displayed together with the data.

continues on next page

Table 7.5 – continued from previous page

Parameter	Type	Description
data_capabilities[].contacts	JSON Object Array	OPTIONAL. Array containing customer service contacts or user support channels related to the specific dataset. Each object contains: • **type**: REQUIRED. Indicates the type of contact information and MUST be set to `email`, `telephone` or `url`. • **value**: REQUIRED. Contains the value of the contact information corresponding to the specified type. • **description**: OPTIONAL. A textual description associated with the contact information.

Note

For further details on the required features and the expected outcome in terms of user experience, see the Section *Issuance from the Wallet Instance Catalog* (page 43) for the parameter *data_capabilities.user_information* and Section *Focus on Electronic Attestations of Attributes* (page 55) for the parameters *organization_info.logo_uri*, *organization_info.logo_extended_uri*, *data_capabilities.logo_uri*, *data_capabilities.background_color* and *data_capabilities.available_claims.order*.

AS Registry Example

A non-normative example of AS Registry structure is given below:

```
{
    "id": "urn:authentic-sources:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "localization": {
        "default_locale": "it",
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/",
        "version": "1.0.0"
    },
    "authentic_sources": [
        {
```

```
            "entity_id": "https://motorizzazione.gov.example",
            "organization_info": {
                "contacts": [
                    "registry@motorizzazione.gov.example",
                    "technical-support@motorizzazione.gov.example"
                ],
                "dpa_contact": "dpa@motorizzazione.gov.example",
                "homepage_uri": "https://www.gov.example/transport",
                "ipa_code": "m_inf",
                "legal_identifier": "80192770587",
                "logo_extended_uri": "https://trust-registry.eid-wallet.example.it/logos/m_inf/
extended-image.svg",
                "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3
fff1fa07e998e86f7f7a27ae3",
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/m_inf/image.svg
",
                "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e
998e86f7f7a27ae3",
                "organization_country": "IT",
                "organization_name": "Ministero dei Trasporti - Direzione generale per la Motor
izzazione",
                "organization_name_l10n_id": "authentic_source1.name",
                "organization_type": "public",
                "policy_uri": "https://www.gov.example/transport/privacy-policy",
                "tos_uri": "https://www.gov.example/transport/terms-of-service"
            },
            "data_capabilities": [
                {
                    "api_specification": "https://docs.gov.example/transport/api-oas3.yaml",
                    "available_claims": [
                        {
                            "claim_name": "given_name",
                            "mandatory": true,
                            "order": 1
                        },
                        {
                            "claim_name": "family_name",
                            "mandatory": true,
                            "order": 2
                        },
                        {
                            "claim_name": "birth_date",
                            "mandatory": true,
                            "order": 3
                        },
                        {
                            "claim_name": "birth_place",
```

```
        "mandatory": true,
        "order": 4
    },
    {
        "claim_name": "issue_date",
        "mandatory": true,
        "order": 5
    },
    {
        "claim_name": "document_iss_country",
        "mandatory": true,
        "order": 6
    },
    {
        "claim_name": "document_iss_authority",
        "mandatory": true,
        "order": 7
    },
    {
        "claim_name": "expiry_date",
        "mandatory": true,
        "order": 8
    },
    {
        "claim_name": "document_number",
        "mandatory": true,
        "order": 9
    },
    {
        "claim_name": "portrait",
        "mandatory": true,
        "order": 10
    },
    {
        "claim_name": "un_distinguishing_sign",
        "mandatory": true,
        "order": 11
    },
    {
        "claim_name": "restrictions_conditions",
        "mandatory": false,
        "order": 12
    },
    {
        "claim_name": "driving_privileges",
        "mandatory": true,
        "order": 13
```

```
                    }
                ],
                "background_color": "#12107c",
                "contacts": [
                    {
                        "type": "email",
                        "value": "citizen-support@motorizzazione.gov.example"
                    }
                ],
                "data_origin_l10n_id": "authentic_source1.dataset1.origin",
                "data_provision": {
                    "deferred_flow": false,
                    "immediate_flow": true
                },
                "dataset_id": "38832801",
                "integration_endpoint": "https://api.gov.example/transport/driving-license"
,
                "integration_method": "pdnd",
                "intended_purposes": [
                    "IDENTITY_VERIFICATION",
                    "PERSON_IDENTIFICATION",
                    "DRIVING_RIGHTS_VERIFICATION",
                    "LAW_ENFORCEMENT_CONTROLS"
                ],
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38832801/im
age.svg",
                "logo_uri#integrity": "sha-256-b1946ac92492d2347c6235b4d2611184e2a3f8b8c8f8
f8e8f8f8f8f8f8f8f8f8",
                "service_documentation_uri": "https://docs.gov.example/transport/api-docs",
                "update_frequency": "real_time",
                "user_information_l10n_id": "authentic_source1.dataset1.userinfo"
            }
        ]
    },
    {
        "entity_id": "https://api.bank.example/auth-source",
        "organization_info": {
            "contacts": [
                "digital-credentials@bank.example",
                "api-support@bank.example"
            ],
            "dpa_contact": "dpa@bank.example",
            "homepage_uri": "https://www.bank.example",
            "legal_identifier": "12345678901",
            "logo_extended_uri": "https://trust-registry.eid-wallet.example.it/logos/123456
78901/extended-image.svg",
```

```
            "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3
fff1fa07e998e86f7f7a27ae3",
            "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/12345678901/ima
ge.svg",
            "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e
998e86f7f7a27ae3",
            "organization_country": "IT",
            "organization_name": "Banca Esempio SpA",
            "organization_name_l10n_id": "authentic_source2.name",
            "organization_type": "private",
            "policy_uri": "https://www.bank.example/privacy-policy",
            "tos_uri": "https://www.bank.example/terms-of-service"
        },
        "data_capabilities": [
            {
                "api_specification": "https://api.bank.example/docs/psd2-openapi.yaml",
                "available_claims": [
                    {
                        "claim_name": "given_name",
                        "mandatory": true,
                        "order": 1
                    },
                    {
                        "claim_name": "family_name",
                        "mandatory": true,
                        "order": 2
                    },
                    {
                        "claim_name": "birth_date",
                        "mandatory": false,
                        "order": 3
                    },
                    {
                        "claim_name": "tax_id_code",
                        "mandatory": true,
                        "order": 4
                    },
                    {
                        "claim_name": "account_holder_name",
                        "mandatory": true,
                        "order": 5
                    },
                    {
                        "claim_name": "iban",
                        "mandatory": true,
                        "order": 6
                    },
```

```
                    {
                        "claim_name": "account_status",
                        "mandatory": false,
                        "order": 7
                    },
                    {
                        "claim_name": "account_opening_date",
                        "mandatory": false,
                        "order": 8
                    },
                    {
                        "claim_name": "bank_name",
                        "mandatory": true,
                        "order": 9
                    },
                    {
                        "claim_name": "bank_code",
                        "mandatory": true,
                        "order": 10
                    }
                ],
                "background_color": "#003366",
                "contacts": [
                    {
                        "type": "email",
                        "value": "digital-credentials@bank.example"
                    }
                ],
                "data_origin_l10n_id": "authentic_source2.dataset1.origin",
                "data_provision": {
                    "deferred_flow": false,
                    "immediate_flow": true
                },
                "dataset_id": "38854801",
                "integration_endpoint": "https://api.bank.example/psd2/v1/accounts",
                "integration_method": "pdnd",
                "intended_purposes": [
                    "PAYMENT_AUTHORIZATION",
                    "ELIGIBILITY_VERIFICATION"
                ],
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38854801/im
age.svg",
                "logo_uri#integrity": "sha-256-d8e8fca2dc0f896fd7cb4cb0031ba249e3f8b8c8f8f8
e8f8f8f8f8f8f8f8f8f9",
                "service_documentation_uri": "https://docs.bank.example/psd2-api",
                "update_frequency": "daily",
                "user_information_l10n_id": "authentic_source2.dataset1.userinfo"
```

```
    },
    {
        "api_specification": "https://api.bank.example/docs/psd2-openapi.yaml",
        "available_claims": [
            {
                "claim_name": "given_name",
                "mandatory": true,
                "order": 1
            },
            {
                "claim_name": "family_name",
                "mandatory": true,
                "order": 2
            },
            {
                "claim_name": "tax_id_code",
                "mandatory": true,
                "order": 3
            },
            {
                "claim_name": "transaction_history",
                "mandatory": true,
                "order": 4
            },
            {
                "claim_name": "average_balance",
                "mandatory": false,
                "order": 5
            },
            {
                "claim_name": "credit_score_indicator",
                "mandatory": false,
                "order": 6
            }
        ],
        "background_color": "#003366",
        "contacts": [
            {
                "type": "email",
                "value": "digital-credentials@bank.example"
            }
        ],
        "data_origin_l10n_id": "authentic_source2.dataset2.origin",
        "data_provision": {
            "deferred_flow": true,
            "immediate_flow": false,
            "max_response_time_minutes": 1440,
```

```
                        "notification_methods": [
                            "push",
                            "poll"
                        ]
                    },
                    "dataset_id": "38854802",
                    "integration_endpoint": "https://api.bank.example/psd2/v1/transactions",
                    "integration_method": "pdnd",
                    "intended_purposes": [
                        "PROOF_OF_PAYMENT",
                        "RECURRING_PAYMENT_MANAGEMENT"
                    ],
                    "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38854802/image.svg",
                    "logo_uri#integrity": "sha-256-f4a2b7e1dc0f896fd7cb4cb0031ba249e3f8b8c8f8f8e8f8f8f8f8f8f8f8fb",
                    "service_documentation_uri": "https://docs.bank.example/psd2-api",
                    "update_frequency": "weekly",
                    "user_information_l10n_id": "authentic_source2.dataset2.userinfo"
                }
            ]
        }
    ]
}
```


Note

For a better and more efficient management of the localization of the information contained in the Authentic Source Registry, an Entity consulting it SHOULD:

- Download the basic version of the Authentic Source Registry (compact, without localizations) using the `.well-known/authentic-sources` endpoint.
- Determine the User's preferred language.
- Download only the necessary localization bundles.
- Dynamically merge localised content with the Authentic Source Registry structure.

A non-normative example of a localization bundle output is given below:


```
{
  "authentic_source1.name": "Ministero delle infrastrutture e dei trasporti",
  "authentic_source1.dataset1.origin": "MIT -- Direzione Generale per la Motorizzazione",
  "authentic_source1.dataset1.userinfo": "###### Patente di Guida\nSono disponibili le patenti rilasciate dopo il 1° gennaio 2020. Per le patenti più vecchie, contattare l'ufficio motorizzazione locale.",
```

```
  "authentic_source2.name": "Banca Esempio SpA",
  "authentic_source2.dataset1.origin": "Esempio origine dei dati 1",
  "authentic_source2.dataset1.userinfo": "###### Informazioni sulla disponibilità dei dati\nL'a
ccesso ai dati finanziari richiede il consenso del cliente ed è soggetto alla normativa PSD2. L
e informazioni sui conti sono disponibili solo per i conti attivi.",
  "...": "..."
}
```

Localization bundles MUST be available at the URI composed by appending the locale code and `.json` to the `localization.base_uri` value defined in the registry. Each locale bundle MUST be accessible following the naming pattern **{locale_code}.json**, where **{locale_code}** is replaced with the corresponding locale code from the **available_locales** array.

A non-normative example of the Italian localization URI for the bundle would be **https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/it.json**.

7.4.4 AS-CI Coordination

Following AS registration, the AS Registry enables Credential Issuers to discover suitable AS entities and request integration approval. This coordination process is detailed in *Authentic Source to Credential Issuer Authorization Process* (page 240).

7.5 Federation Registry

The **Federation Registry** provides the cryptographic trust infrastructure for all IT-Wallet ecosystem participants. The Federation Registry maintains the authoritative list of trusted entities and their operational status using federation-specific endpoints as defined in *Federation API endpoints* (page 114).

7.5.1 Registry Integration Role

Within the IT-Wallet System Register architecture, the Federation Registry serves as the **trust validation layer** for:

1. **Entity Authentication**: Validates the cryptographic identity of all participants before registry operations
2. **Trust Chain Verification**: Provides the cryptographic foundation for Credential Issuers, Relying Parties, and Wallet Providers entity validation
3. **Compliance Verification**: Maintains Trust Marks that attest regulatory compliance and operational status

7.5.2 Federation Registry Access

Federation Registry operations are accessed through the Trust Anchor's federation endpoints as detailed in *Federation API endpoints* (page 114). The registry discovery architecture provides federation endpoint information via the registry discovery endpoint described in *Registry Discovery Endpoint* (page 150).

Note

Federation endpoints are available through both the registry discovery mechanism (for unified registry access) and the Trust Anchor's Entity Configuration at `.well-known/openid-federation` (for federation-specific operations). Both sources provide the same endpoint URLs but serve different discovery patterns: registry discovery for initial ecosystem orientation, Entity Configuration for standard OpenID Federation 1.0 compliance.

For complete technical specifications of federation protocols, entity configurations, trust evaluation mechanisms, and trust chain validation, see *The Infrastructure of Trust* (page 109).

7.6 Digital Credentials Catalog

The Digital Credentials Catalog is the registry of all available Digital Credentials recognized within the IT-Wallet ecosystem. It is published by the Trust Anchor and publicly available by all Entities through a specialized Federation endpoint. It acts as a single reference point for all actors involved in the process of issuing, verifying and using Digital Credentials.

The Digital Credential Catalog aims to:

1. Facilitate Digital Credential discovery for Users.
2. Standardize the technical and functional description of Digital Credentials.
3. Enable interoperability between different Issuers and Relying Parties.
4. Simplify the integration process for Wallet Providers and Relying Parties.
5. Ensure trust in the ecosystem through verifiable and trustworthy information.
6. Provide transparency on the ecosystem of available Digital Credentials.

The main Entities involved in the Digital Credential Catalog are:

- **Trust Anchor**: It manages and maintains the Digital Credential Catalog, guaranteeing its authenticity and integrity.
- **Supervisory Body**: It interacts with the Trust Anchor and the Digital Credential Catalog to monitor the registration phase ensuring security and privacy according to national/European regulations, keeping all the information reliable and updated.

- **Digital Credential Issuers**: The entities authorized to issue Digital Credentials, registering them in the Catalog.
- **Relying Parties**: They use the Digital Credential Catalog to gather all the information needed about the Digital Credentials they intend to request during the presentation phase.
- **Wallet Providers**: They access the Digital Credential Catalog to identify the available Digital Credentials and to retrieve all necessary information for integrating them into their Wallet Solutions.
- **Users**: The Users who indirectly use the Digital Credentials Catalog through their Wallet Instances to discover and request Digital Credentials.
- **Authentic Sources**: The Entities that hold the original data that is attested in the Digital Credentials. They provide support to Issuers in registering the Digital Credentials in the Catalog.

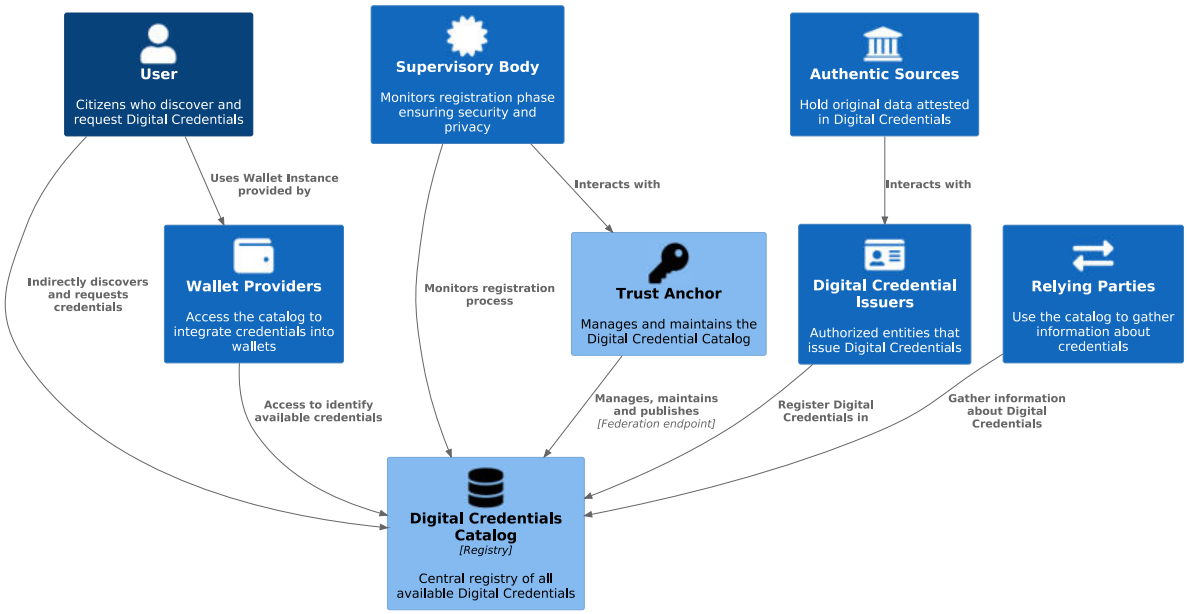


Fig. 7.1: Entity-Relationship diagram of Digital Credential Catalog.

The following table summarizes the main information that MUST be provided by the Digital Credential Catalog:

Table 7.6: Digital Credential Catalog - Main information

Information related to	Description
Digital Credential Metadata	Essential identifying information and characteristics of the Digital Credential, including: • **Credential Unique identifier**: A unique identifier string of each Digital Credential. • **User authentication methods**: User authentication mechanisms used to request the Digital Credential, if required by Issuers or Authentic Sources. • **Minimum Level of Assurance**: The minimum Level of Assurance required for the Digital Credential's reliability. It MUST take into account the Level of Assurance of User authentication, when applicable, and Wallet Instance.
Digital Credential Issuers	Details about the organization authorized to issue the Digital Credential, such as: • **Issuer identifiers**: Unique identifier for the Digital Credential issuer. • **Issuer type**: Classification as PID, (Q)EAA, or Pub-EAA Provider. • **Additional information**: Organizational details including name, code, and contact information.
Authentic Sources Technical Specification	Information about the authoritative data source. Technical details, including: • **Digital Credential schemes**: Framework and structure specifications. • **Digital Credential formats**: Data format and encoding standards. • **Authentication policy**: Methods and requirements for verification.

continues on next page

Table 7.6 – continued from previous page

Information related to	Description
Terms of Use	Conditions and limitations for Digital Credential usage, such as: • **Credential validity**: Time period during which the Digital Credential is valid and, when applicable, mechanisms and technical details for invalidating Digital Credentials (revocation/suspension methods). • **Restriction policy**: If applicable, rules governing the Digital Credential's use and limitations according to national regulations. It is used, for example, to specify if only specific legal type Entities, for example Pub-EAA Provider and public Wallet Solutions, are allowed to issue and obtain the Digital Credential. • **Pricing policy**: Information related to pricing models of Digital Credential, such as *free*, *issuance_based*, *verification_based*. • **Digital Credential purposes**: Information related to the allowed purposes for which the Digital Credential can be used. Each Digital Credential type can be used for multiple purposes.

The Trust Anchor MUST publish and keep up to date all the information at the Digital Credential Catalog *.well-known* endpoint ensuring data reliability, authenticity and integrity. In particular, the Digital Credential Catalog MUST be available through the `.well-known/credential-catalog` endpoint. It MUST support `application/jose` as content-type.

Below a non-normative example is given.

```
GET /.well-known/credential-catalog HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/jose

HTTP/1.1 200 OK
Content-Type: application/jose

eyJhbGciOiJSUzI1NiIsImtpZCI6ImV4YW1w...
```

In the section *Digital Credentials Catalog Structure* (page 194) an example of Digital Credentials Catalog is given as decoded in JSON.

7.6.1 Digital Credentials Hierarchy

Digital Credentials recognized within the IT-Wallet ecosystem are classified and standardized according to the following multi-level hierarchical model designed to improve semantic clarity, credential discovery, and compatibility with both credential-specific and claim-based verification workflows.

The hierarchy is defined as follows:

Domain

A **Domain** represents a high-level thematic area grouping Credential families that relate to the same broad context (e.g., Identity, Health, Education, Mobility). Domains provide a top-level organizational layer.

Credential Class

A **Credential Class** represents a family of Credentials sharing similar nature, function, or structure (e.g., Identification Documents, Civil Status Certificates).

Each Class SHOULD define:

- a stable Class identifier (URI),
- the expected semantics of the Credential Family.

Classes enable Relying Parties and Wallet Solutions to request or match Credentials based on their type category.

Credential Type

A **Credential Type** represents a specific Credential within a Class (e.g. Digital Travel Credential, Birth Certificate, Mobile Driving License). Each Credential Type MUST include:

- a unique identifier,
- the Credential Issuer identifier,
- the set of Attributes that may be included in presentations.

Credential Types enable precise targeting for compliance-driven or regulation-mandated verification flows.

Purpose (Verification Intent)

A **Purpose (Verification Intent)** describes *why* a credential may be requested by a Relying Party (e.g., Identity Verification, Age Verification, Eligibility for specific services). Purposes MUST describe **verification outcomes**. Each Credential Type MUST declare its Domain, Class, and supported Purposes.

The following tables provide non-exhaustive examples illustrating the relationships between Domains, Credential Classes, and Credential Types, followed by their mapping to verification

Purposes. Additional Domains, Classes, specific Credentials, and verification Purposes **MAY** be added over time as the IT-Wallet ecosystem evolves.

Table 7.7: Digital Credential Taxonomy: Hierarchy and Classification

Domain	Description	Credential Class	Credential Type
IDENTITY	Credentials that establish or confirm a person's legal identity and personal, civil or legal status.	• Identification Documents • Civil Registry and Personal Status Certificates • Economic and Legal Status	• Digital Travel Credential • Mobile Driving License (Italy only) • Tax Code / Health Insurance Card • Age Certification • Birth Certificate • Residence Certificate • Family Status Certificate • Marriage Certificate • Citizenship Certificate • ISEE (Equivalent Economic Situation Indicator) • Residence Permit • Certificate of Pending Charges • Criminal Record Certificate

continues on next page

Table 7.7 – continued from previous page

Domain	Description	Credential Class	Credential Type
HOME AND FAMILY	Credentials that attest household composition, residence, and housing-related legal or fiscal relationships.	• Property and Cadastral Documents • Family Documents • Local Tax Documents	• Deed of Sale • Cadastral Survey • Cadastral Floor Plan • Cadastral Certificate • Children's Tax Code / Health Card • Birth Certificate • Family Status Certificate • IMU (Property Tax) • TARI (Waste Tax)
EDUCATION	Credentials that attest educational achievements, academic qualifications, and professional training.	• Educational Qualifications • Professional Certifications	• Lower Secondary School Diploma • Upper Secondary School Diploma • Bachelor's Degree • Master's Degree • University Master • PhD • Professional Licenses (e.g. architect, lawyer) • Vocational Training Certificates • Language Certifications (e.g. IELTS) • Academic Qualifications (e.g. Europass)

continues on next page

Table 7.7 – continued from previous page

Domain	Description	Credential Class	Credential Type
HEALTH	Credentials related to healthcare coverage, medical status, and health-related certifications.	• Certifications and Eligibility • Medical Records	• Health Insurance Card (TEAM) • European Health Card (CED) • Disability Certificate • Vaccination Certificate • Sports Fitness Certificate • Work Fitness Certificate • Medical Prescriptions • Digital Medical Report
FINANCIAL	Credentials related to payment instruments, financial authorizations, and proof of payments.	• Payment Instruments • Payment Credentials and Authorisations • Public Payments and Fees • Recurring Payments and Subscriptions	• Digital Payment Card (debit / credit / prepaid) • Virtual Card • Bank Account (IBAN) • Strong Customer Authentication (SCA) Credential • Payment Receipt • Digital Stamp Duty (Bollo digitale) • Tax and Fee Payment Certificate • Subscription Mandate • Recurring Payment Credential

continues on next page

Table 7.7 – continued from previous page

Domain	Description	Credential Class	Credential Type
CULTURE AND LEISURE	Credentials that attest membership, affiliation, or participation in cultural or recreational programs.	• Cultural Cards and Benefits • Membership and Loyalty Programs	• Culture Card • Annual Museum Passes • Cinema Card • Museum Card • Association Membership Cards • Library Card • City Pass
EMPLOYMENT	Credentials that attest employment relationships, professional status, and contribution records.	• Employment Documents • Employment Status • Employment Affiliation	• Digital Employment Contract • Curriculum Vitae (CV) • Residence Permit • Employment Status Certificate • INPS Contribution Record • Physical Access Badge

continues on next page

Table 7.7 – continued from previous page

Domain	Description	Credential Class	Credential Type
MOBILITY AND TRAVEL	Credentials that attest mobility rights, vehicle-related status, and travel-related entitlements.	• Licenses and Authorizations • Vehicle Documents • Transport Subscriptions • Travel Documents • Travel Insurance • Bookings • Discounts and Benefits	• Mobile Driving License • Boating License • Vehicle Registration Certificate • Digital RCA Insurance • Vehicle Inspection Certificate • Green Card / International Insurance • Public Transport Pass • Road Charging Subscription • Digital Travel Credential • Travel Tickets (air, train, etc.) • Travel Insurance Policy • Hotel Reservation • Discount Cards • Tourist Benefits

continues on next page

Table 7.7 – continued from previous page

Domain	Description	Credential Class	Credential Type
BONUSES	Credentials that attest entitlement to economic benefits, incentives, or vouchers.	• Economic Benefits and Allowances • Incentives and Vouchers • Health and Wellbeing Bonuses	• Family Allowance Credential • Unemployment Benefit Credential • Digital Voucher • Purchase Incentive Credential • Cashback Eligibility Credential • Healthcare Bonus Credential • Mental Health Support Voucher • Sports and Physical Activity Bonus

Table 7.8: Table 2: Mapping between Credential Classes and Purposes

Credential Class	Supported Purposes
Identification Documents	• Identity verification • Age verification • Person identification
Civil Registry and Personal Status Certificates	• Civil status verification • Right of residence • Household composition verification

continues on next page

Table 7.8 – continued from previous page

Credential Class	Supported Purposes
Economic and Legal Status	• Eligibility for services or benefits • Legal status verification • Criminal record check
Property and Cadastral Documents	• Residence and household verification • Property ownership verification • Real estate compliance
Family Documents	• Household composition verification • Eligibility for family-based social services
Local Tax Documents	• Compliance with local tax obligations • Verification of property tax status
Educational Qualifications	• Qualification and degree verification • Eligibility for education pathways
Professional Certifications	• Professional license verification • Skills assessment for work
Certifications and Eligibility	• Verification of vaccination status • Verification of fitness status • Access to health-restricted areas

continues on next page

Table 7.8 – continued from previous page

Credential Class	Supported Purposes
Medical Records	• Access to healthcare services • Sharing of medical records • Medical history validation
Payment Instruments	• Payment authorization • Payment execution • Proof of payment
Payment Credentials and Authorisations	• Management of financial authorizations • Strong Customer Authentication (SCA)
Public Payments and Fees	• Proof of tax payment • Proof of fee payment • Digital stamp duty validation
Recurring Payments and Subscriptions	• Management of recurring payments • Subscription mandate verification
Cultural Cards and Benefits	• Access to cultural services • Access to leisure services • Application of member discounts
Membership and Loyalty Programs	• Verification of affiliation • Verification of participation • Use of loyalty benefits

continues on next page

Table 7.8 – continued from previous page

Credential Class	Supported Purposes
Employment Documents	• Employment status verification • Professional profile validation
Employment Status	• Verification of contribution records • Eligibility for employment-related benefits
Licenses and Authorizations	• Driving rights verification • Navigation rights verification • Law enforcement controls
Vehicle Documents	• Vehicle registration verification • Vehicle inspection verification • Insurance status check
Transport Subscriptions	• Access to transport services • Public transport pass verification
Travel Documents	• Right to travel or circulate • Cross-border mobility identity check
Travel Insurance and Bookings	• Verification of travel insurance coverage • Accommodation reservation check • Transport reservation check
Discounts and Benefits	• Application of member discounts • Access to tourist benefits

continues on next page

Table 7.8 – continued from previous page

Credential Class	Supported Purposes
Economic Benefits and Allowances	• Eligibility verification for family benefits • Eligibility verification for unemployment benefits • Allocation of economic support
Incentives and Vouchers	• Use of digital vouchers • Use of purchase incentives • Cashback eligibility verification
Health and Wellbeing Bonuses	• Access to healthcare bonuses • Use of mental health vouchers • Use of sports vouchers
Employment Affiliation	• Access permit verification

Each Credential MUST specify domains, classes and purposes to enable both **Credential-Specific Scenarios** and **Credential-Agnostic Scenarios** according to Relying Party's requirements and presentation request patterns, as defined in the mapping tables above.

1. **Credential-Specific Scenarios** (Primary for Government/Regulated Sectors): RPs request specific Credential types for compliance and audit requirements, including for example:

- **Government Services**: `"credential_type":"pid"` for PID-specific identity verification or `"credential_type":"eid"` for IT-Wallet ID-specific identity verification.
- **Police Controls**: `"credential_type":"mDL"` for driving license verification.
- **Banking KYC**: Specific credential types mandated by financial regulations.
- **Healthcare Services**: `"credential_type":"european_disability_card"` for EU-compliant disability benefit access.

2. **Credential-Agnostic Scenarios** (Typical for Private Business): RPs request specific claims regardless of Credential source for operational efficiency, such as:

- **E-commerce Delivery**: Any credential, among those to which he is authorized to access, containing `given_name`, `family_name`, `address` for shipping.
- **Subscriptions**: Any credential, among those to which he is authorized to access, with `given_name`, `email` for personalization.
- **Service Personalization**: Business applications requiring basic personal data without strong source requirements.

This approach allows:

- **Policy-based authorization** by using **Domain / Class / Credential Type / Purpose** mappings.
- **Flexible RP registration** supporting both government compliance needs and business operational requirements.

7.6.2 Digital Credentials Catalog Structure

Digital Credentials Catalog contents is secured in a JWS that contains the following JOSE header parameters:

JOSE header	Description	Reference
typ	REQUIRED. It MUST be set to `JOSE`.	[**RFC 7515** Section 4.1. 9].
alg	REQUIRED. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or with a symmetric algorithm (MAC) identifier.	[**RFC 7515** Section 4.1. 1].
kid	REQUIRED. Unique identifier of the public key.	[**RFC 7515** Section 4.1. 4].
x5c	OPTIONAL. Contains the X.509 public key Certificate or Certificate chain [**RFC 5280**] corresponding to the key used to digitally sign the JWS. When the header parameter *kid* value is present, it MUST refer to the same leaf's cryptographic public key used with the X.509 Certificate.	[**RFC 7515** Section 4.1. 6.].
cty	REQUIRED. It MUST be set to `application/json`.	[**RFC 7515** Section 4.1. 6.].

The JWS payload contains the following parameters:

Table 7.10: First-level Fields of the Digital Credentials Catalog

Field Name	Description
id	REQUIRED. Unique identifier of the Digital Credentials Catalog (e.g., `urn:credential-catalog:it-wallet`).
version	REQUIRED. Version of the Digital Credential Catalog format.
last_modified	REQUIRED. Timestamp of the last modification to the Digital Credential Catalog (e.g., `2025-03-15T12:00:00Z`).
iss	REQUIRED. Issuer identifier of the Digital Credential Catalog.
localization	REQUIRED. Localization configuration object containing: • **default_locale**: Default locale code (e.g., `it`). • **available_locales**: Array of supported locale codes (e.g., `["en", "it"]`). • **base_uri**: Base URI for localization bundle retrieval (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog/`). • **version**: Version of the localization bundle format.
credentials	REQUIRED. Array containing Digital Credential definitions.

Each element of the `credentials` array contains at least the following information:

Table 7.11: First-level Fields of Each Credential Entry

Field Name	Description
version	REQUIRED. Version of the Digital Credential definition.
credential_type	REQUIRED. Unique identifier of the Digital Credential type. For PID it MUST be `pid` and for IT-Wallet ID MUST be `eid`.
credential_name_l10n_id	REQUIRED. Localization key referencing the human-readable name of the Digital Credential in the localization bundle (e.g., `mDL.name`).
legal_type	REQUIRED. Legal classification of the Credential (e.g., `pub-eaa`, `qeaa`, `eaa`).

continues on next page

Table 7.11 – continued from previous page

Field Name	Description
restriction_policy	OPTIONAL. Legal restrictions on Wallet Solutions and/or Credential Issuers allowed to request/issue the Digital Credential. • **allowed_wallet_ids**: List of allowed Wallet Solutions identifiers. • **allowed_issuer_ids**: List of allowed Credential Issuers identifiers. If present, it represents a whitelist of Credential Issuers that may be added by the Trust Anchor in the **issuers** field of the corresponding Digital Credential. • **presentation_flows**: Type of presentation flows supported; remote and/or proximity flow.
pricing_policy	OPTIONAL. Information about Digital Credential pricing, including: • **models**: REQUIRED. Array of pricing models applicable to the Digital Credential, each containing – **pricing_type**: Type of pricing model, such as `issuance_based`, `verification_based`, `subscription_based`, `other`. – **price**: Cost associated with the model. – **currency**: Currency of the price. • **pricing_model_uri**: URI to the detailed pricing model documentation.
validity_info	REQUIRED. Information about Digital Credential validity, including at least: • **max_validity_days**: Maximum validity period in days. • **status_methods**: Supported status verification methods (e.g. `status_list`). • **allowed_states**: Array of objects representing allowed Digital Credential states. Each object contains a hex status code (e.g. , `0x00` for `VALID`, `0x01` for `INVALID`, `0x02` for `SUSPENDED`, `0x03` for `UPDATE`, `0x0F` for `ATTRIBUTE_UPDATE`), a `title_l10n_id` localization key, and a `description_l10n_id` localization key for UI display. • **administrative_expiration_user_info**: OPTIONAL. Object containing `title_l10n_id` and `description_l10n_id` keys for displaying administrative expiration information to the User.

continues on next page

Table 7.11 – continued from previous page

Field Name	Description
authentication	REQUIRED. Digital Credential authentication requirements. • **user_auth_required**: REQUIRED. Flag indicating if User authentication is required during the issuance of the Digital Credential. • **min_loa**: REQUIRED. Minimum Level of Assurance required for Digital Credential authentication. It MUST include the Level of Assurance of the User authentication and the Wallet Instance requesting the Digital Credential. • **supported_schemes**: REQUIRED if `user_auth_required` is `true`. Supported digital identity authentication schemes (e.g., `["it_wallet"]`).
domains	REQUIRED. Array of domain IDs to which Digital Credential belongs (e.g., `"IDENTITY"`, `"MOBILITY_TRAVEL"`).
classes	REQUIRED. Array of class IDs to which Digital Credential belongs (e.g., `"IDENTIFICATION_DOCUMENTS"`, `"LICENSES_AUTHORIZATIONS"`).
purposes	REQUIRED. Array of usage purpose IDs for which the Digital Credential can be used, defining specific usage contexts and required claims for each purpose (e.g., `"IDENTITY_VERIFICATION"`, `"AGE_VERIFICATION"`, `"DRIVING_RIGHTS_VERIFICATION"`).

continues on next page

Table 7.11 – continued from previous page

Field Name	Description
issuers	REQUIRED. Array of relevant information about authorized Credential Issuers, including administrative and technical data such as Organization name, a reference to the API specification document and supported issuance mechanisms. Each array element contains: • **entity_id**: REQUIRED. String. Unique identifier of the Credential Issuer. It MUST match with the value contained in the `iss` parameter of the Credential Issuer Entity Configuration. • **organization_name_l10n_id**: REQUIRED. String. Localization key referencing the localized organization name in the localization bundle (e.g., `issuer1.name`). • **organization_code**: REQUIRED. String. Credential Issuer IPA code for government entities or VAT number for private entities. • **organization_country**: REQUIRED. String. Two-letter ISO 3166-1 alpha-2 country code of the organization. • **contacts**: REQUIRED. String. Array of contact email addresses for at least one user-support, one application, and one systems specialist. • **legal_type**: REQUIRED. String. Legal classification of the Credential Issuer (e.g., pub-eaa, qeaa, eaa). • **homepage_uri**: REQUIRED. String. URL pointing to the organization's homepage. • **logo_uri**: OPTIONAL. String. URL to the organization's logo image. • **policy_uri**: REQUIRED. String. URL to privacy policy document. • **tos_uri**: OPTIONAL. String. URL to terms of service document. • **service_documentation_uri**: OPTIONAL. String. URL pointing to the Credential Issuer service documentation. • **issuance_flows**: REQUIRED. Object. It contains the following parameters: – **deferred_flow**: REQUIRED. Boolean. Indicates if the deferred issuance is supported. – **immediate_flows**: REQUIRED. Boolean. Indicates if the immediate issuance is supported. – **wallet_initiated**: REQUIRED. Boolean. Indicates if the Wallet-Initiated flow is supported. – **issuer_initiated**: REQUIRED. Boolean. Indicates if the Issuer-Initiated flow issuance is supported (Third Party Initiated Flow). – **max_deferred_issuance_time_minutes**: CONDITIONAL. Integer. Maximum time in minutes for the availability of the issuance of the Credential. REQUIRED

Table 7.11 – continued from previous page

Field Name	Description
authentic_sources	CONDITIONAL. It is REQUIRED only if `parent_credentials` is absent. Array of Authentic Source JSON objects referencing authorized Authentic Sources. Each object MUST contain the AS entity identifier and the specific data capability identifier: • **id**: String identifier referencing the Authentic Source entity_id as registered in the *Authentic Source Registry* (page 163). • **dataset_id**: String identifier of the specific data capability/ dataset used by the Issuer from the AS.
parent_credentials	CONDITIONAL. It is REQUIRED only if `authentic_sources` is absent. Array of `credential_type` identifier corresponding to Credentials designated as data sources. Each element identifies a Credential that acts as an Authentic Source during the Digital Credential Issuance process.

Note

The union of `credential_type` and `version` MUST be unique in the Credential Catalog.

The corresponding example of Digital Credentials Catalog as decoded in JSON for both header and payload is the following:

```
{
    "typ":"JOSE",
    "alg":"ES256",
    "kid":"e9bc097a-ce51-4036-9562-d2ade882db0d",
    "cty":"application/json"
}
```

```
{
    "id": "urn:credential-catalog:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "iss": "https://trust-registry.eid-wallet.example.it",
    "localization": {
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog",
        "default_locale": "it",
        "version": "1.0.0"
```

```
    },
    "credentials": [
        {
            "version": "1",
            "credential_type": "mDL",
            "credential_name_l10n_id": "mDL.name",
            "legal_type": "pub-eaa",
            "restriction_policy": {
                "allowed_wallet_ids": [
                    "https://wallet-provider.example.org/wallet_solution",
                    "https://wallet-provider2.example.org/wallet_solution"
                ],
                "allowed_issuer_ids": [
                    "https://issuer.example.org"
                ],
                "presentation_flows": {
                    "remote": true,
                    "proximity": true
                }
            },
            "pricing_policy": {
                "models": [
                    {
                        "pricing_type": "verification_based",
                        "price": 0.01,
                        "currency": "EUR"
                    }
                ],
                "pricing_model_uri": "https://example.com/pricing"
            },
            "validity_info": {
                "max_validity_days": 365,
                "status_methods": [
                    "status_list"
                ],
                "administrative_expiration_user_info": {
                    "title_l10n_id": "mDL.administrative_expiration_user_info.title",
                    "description_l10n_id": "mDL.administrative_expiration_user_info.description
"
                },
                "allowed_states": [
                    {
                    "0x00": "VALID",
                    "title_l10n_id": "mDL.VALID.title",
                    "description_l10n_id": "mDL.VALID.description"
                    },
                    {
```

```
            "0x01": "INVALID",
            "title_l10n_id": "mDL.INVALID.title",
            "description_l10n_id": "mDL.INVALID.description"
            },
            {
            "0x02": "SUSPENDED",
            "title_l10n_id": "mDL.SUSPENDED.title",
            "description_l10n_id": "mDL.SUSPENDED.description"
            },
            {
            "0x03": "UPDATE",
            "title_l10n_id": "mDL.UPDATE.title",
            "description_l10n_id": "mDL.UPDATE.description"
            },
            {
            "0x0F": "ATTRIBUTE_UPDATE",
            "title_l10n_id": "mDL.ATTRIBUTE_UPDATE.title",
            "description_l10n_id": "mDL.ATTRIBUTE_UPDATE.description"
            }
        ]
    },
    "authentication": {
        "user_auth_required": true,
        "min_loa": "high",
        "supported_schemes": [
            "it_wallet"
        ]
    },
    "domains": ["IDENTITY", "MOBILITY_TRAVEL"],
    "classes": ["IDENTIFICATION_DOCUMENTS", "LICENSES_AUTHORIZATIONS"],
    "purposes": ["IDENTITY_VERIFICATION", "AGE_VERIFICATION", "DRIVING_RIGHTS"],
    "issuers": [
        {
            "entity_id": "https://issuer.example.org",
            "organization_name_l10n_id": "mDL.issuer1.name",
            "organization_code": "ci_example_it",
            "organization_country": "IT",
            "contacts": [
                "mailto:informazioni@example.it",
                "mailto:protocollo@pec.example.it"
            ],
            "legal_type": "pub-eaa",
            "homepage_uri": "https://issuer.example.org",
            "logo_uri": "https://issuer.example.org/logo.svg",
            "policy_uri": "https://issuer.example.org/privacy",
            "tos_uri": "https://issuer.example.org/terms",
```

```
                    "service_documentation_uri": "https://issuer.example.org/.well-known/service-doc",
                    "issuance_flows": {
                        "wallet_initiated": true,
                        "issuer_initiated": true,
                        "deferred_flow": true,
                        "max_deferred_issuance_time_minutes": 1440,
                        "notification_methods": [
                            "push",
                            "polling"
                        ]
                    }
                }
            ],
            "authentic_sources": [
                {
                    "id": "https://motorizzazione.gov.example",
                    "dataset_id": "38832801"
                }
            ]
        }
    ]
}
```


Note

For a better and more efficient management of the localization of the information contained in the Digital Credentials Catalog, an Entity consulting it SHOULD:

- Download the basic version of the Digital Credentials Catalog (compact, without localizations) using the `.well-known/credential-catalog` endpoint.
- Determine the User's preferred language.
- Download only the necessary localization bundles.
- Dynamically merge localised content with the Digital Credentials Catalog structure.

A non-normative example of a localization bundle output is given below:


```
{
  "mDL.name": "Patente di Guida",
  "mDL.issuer1.name": "Esempio di Credential Issuer",
  "...": "..."
}
```


Localization bundles MUST be available at the URI composed by appending the locale code and `.json` to the `localization.base_uri` value defined in the catalog. Each locale bundle MUST

be accessible following the naming pattern **{locale_code}.json**, where **{locale_code}** is replaced with the corresponding locale code from the **available_locales** array.

A non-normative example of the Italian localization URI for the bundle would be **https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog/it.json**.

7.6.3 Decentralization of Display and Claim Information

The canonical source for display characteristics and claim structure is determined by the **Credential Issuer's Metadata (Entity Configuration)**.

The overall logic for presenting a Credential is:

1. The Wallet/Relying Party retrieves the lightweight **Digital Credentials Catalog** to discover the available *credential_type* and the *entity_id* of their Credential Issuers.
2. It retrieves the full **Credential Issuer Metadata** (Entity Configuration) from the discovered *entity_id*.
3. The Credential Issuer Metadata MUST contain the full display characteristics (logos, colors) and the detailed schema information (via links to the appropriate Type Metadata or directly in the configuration). The Issuer builds this metadata based on the suggestions provided by the Authentic Source (via the AS Registry) and the standard schema specifications (via the Schema Registry).

7.7 Taxonomy

The **Taxonomy** provides the semantic foundation for Digital Credential interoperability by maintaining the authoritative vocabulary for organizing Credentials within the IT-Wallet ecosystem. The taxonomy is neutral with respect to the Credential format.

The Taxonomy provides, in a single resource, the hierarchical classification system organizing Domains, Classes and Purposes that can be applied to Credential Types, supporting authorization policy evaluation and ecosystem-wide standardization.

Taxonomy Objectives:

1. **Semantic Foundation**: Establish standardized vocabulary for domains and purposes across the ecosystem
2. **Policy Framework**: Enable structured authorization decisions based on hierarchical classification
3. **Interoperability**: Ensure consistent interpretation of credential classifications
4. **Extensibility**: Support evolution of the ecosystem with new Domains, Classes, Credential Types and Purposes

5. **Cross-Border Compliance**: Align with EU regulatory requirements and international standards

Taxonomy Structure:

The taxonomy maintains a four level hierarchical structure:

- **Domains**: Top-level classification representing broad functional areas (e.g., IDENTITY, HEALTH, FINANCIAL)
- **Class (Credential Family)**: Family of Credentials sharing similar function, structure, or legal meaning (e.g., Identification Documents, Civil Status Certificates, Professional Licenses, Access)
- **Credential Type**: Specific Credential definition issued by an authority (e.g., Digital Travel Credential, Birth Certificate, Mobile Driving License).
- **Purpose (Verification Intent)**: Verification objectives that a Credential can satisfy (e. g., Identity Verification, Age Verification, Eligibility for specific services, Access permit verification).

Note

Credential Type is a concept defined at the Digital Credentials Catalog level, not within the Taxonomy. The Taxonomy provides the classification vocabulary (Domains, Classes, Purposes) that Credential Types in the Catalog reference.

Localization Support:

The taxonomy supports multilingual environments through the `_l10n_id` suffix pattern, enabling efficient localization management for user interfaces and cross-border implementations.

Taxonomy Usage:

- **Claims Registry**: Individual claims catalog
- **AS Registry**: Authentic Sources declare capabilities using taxonomy classifications
- **Digital Credentials Catalog**: Credential Types specify Domains, Classes and Purposes
- **Authorization Policies**: Policy evaluation leverages taxonomy structure for access control decisions

The Taxonomy is accessible through the dedicated taxonomy endpoint as defined in the registry discovery mechanism and is maintained by the Supervisory Body to ensure regulatory compliance and semantic consistency.

Taxonomy JSON Structure:

Table 7.12: First-level Fields of the Taxonomy

Field Name	Description
id	REQUIRED. Unique identifier of the Taxonomy (e.g., `urn:taxonomy:it-wallet`).
version	REQUIRED. The version of the Taxonomy (e.g., `1.0.0`).
last_modified	REQUIRED. The timestamp indicating when the Taxonomy was last updated (e.g., `2026-03-11T00:00:00Z`).
name_l10n_id	REQUIRED. Localization key referencing the human-readable name of the Taxonomy (e.g., `taxonomy.name`).
description_l10n_id	REQUIRED. Localization key referencing the human-readable description of the Taxonomy (e.g., `taxonomy.description`).
localization	REQUIRED. Localization configuration object containing: • **default_locale**: Default locale code (e.g., `it`). • **available_locales**: Array of supported locale codes (e.g., `["en", "it"]`). • **base_uri**: Base URI for localization bundle retrieval (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/`). • **version**: Version of the localization bundle format.
domains	REQUIRED. Array of Domain objects, each containing: • **id**: Unique Domain identifier in SCREAMING_SNAKE_CASE (e.g., `IDENTITY`). • **name_l10n_id**: Localization key for the domain name (e.g., `domain.identity.name`). • **description_l10n_id**: Localization key for the domain description (e.g., `domain.identity.description`). • **classes**: Array of Class objects. Each class contains `id`, `name_l10n_id`, and `supported_purposes` (array of purpose ID strings).
purposes	REQUIRED. Flat array of all Purpose objects defined across the taxonomy, each containing: • **id**: Unique Purpose identifier in SCREAMING_SNAKE_CASE (e.g., `IDENTITY_VERIFICATION`, `ACCESS_PERMIT`). • **name_l10n_id**: Localization key for the purpose name (e.g., `purpose.identity_verification.name`).

A non-normative example of Taxonomy structure is given below:

```
{
    "id": "urn:taxonomy:it-wallet",
```

```
"version": "1.0.0",
"last_modified": "2026-03-11T00:00:00Z",
"name_l10n_id": "taxonomy.name",
"description_l10n_id": "taxonomy.description",
"localization": {
    "available_locales": [
        "en",
        "it"
    ],
    "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/",
    "default_locale": "it",
    "version": "1.0.0"
},
"domains": [
    {
        "classes": [
            {
                "id": "IDENTIFICATION_DOCUMENTS",
                "name_l10n_id": "class.identification_documents.name",
                "supported_purposes": [
                    "IDENTITY_VERIFICATION",
                    "AGE_VERIFICATION",
                    "PERSON_IDENTIFICATION"
                ]
            },
            {
                "id": "CIVIL_REGISTRY_STATUS",
                "name_l10n_id": "class.civil_registry_status.name",
                "supported_purposes": [
                    "CIVIL_STATUS_VERIFICATION",
                    "RIGHT_OF_RESIDENCE",
                    "HOUSEHOLD_COMPOSITION_VERIFICATION"
                ]
            },
            {
                "id": "ECONOMIC_LEGAL_STATUS",
                "name_l10n_id": "class.economic_legal_status.name",
                "supported_purposes": [
                    "ELIGIBILITY_VERIFICATION",
                    "LEGAL_STATUS_VERIFICATION",
                    "CRIMINAL_RECORD_CHECK"
                ]
            }
        ],
        "description_l10n_id": "domain.identity.description",
        "id": "IDENTITY",
```

```
        "name_l10n_id": "domain.identity.name"
    },
    {
        "classes": [
            {
                "id": "PROPERTY_CADASTRAL_DOCS",
                "name_l10n_id": "class.property_cadastral_docs.name",
                "supported_purposes": [
                    "RESIDENCE_HOUSEHOLD_VERIFICATION",
                    "PROPERTY_OWNERSHIP_VERIFICATION",
                    "REAL_ESTATE_COMPLIANCE"
                ]
            },
            {
                "id": "FAMILY_DOCUMENTS",
                "name_l10n_id": "class.family_documents.name",
                "supported_purposes": [
                    "HOUSEHOLD_COMPOSITION_VERIFICATION",
                    "FAMILY_SOCIAL_SERVICES_ELIGIBILITY"
                ]
            },
            {
                "id": "LOCAL_TAX_DOCUMENTS",
                "name_l10n_id": "class.local_tax_documents.name",
                "supported_purposes": [
                    "LOCAL_TAX_COMPLIANCE",
                    "PROPERTY_TAX_VERIFICATION"
                ]
            }
        ],
        "description_l10n_id": "domain.home_family.description",
        "id": "HOME_FAMILY",
        "name_l10n_id": "domain.home_family.name"
    },
    {
        "classes": [
            {
                "id": "EDUCATIONAL_QUALIFICATIONS",
                "name_l10n_id": "class.educational_qualifications.name",
                "supported_purposes": [
                    "QUALIFICATION_DEGREE_VERIFICATION",
                    "EDUCATION_PATHWAY_ELIGIBILITY"
                ]
            },
            {
                "id": "PROFESSIONAL_CERTIFICATIONS",
                "name_l10n_id": "class.professional_certifications.name",
```

```
                "supported_purposes": [
                    "PROFESSIONAL_LICENSE_VERIFICATION",
                    "SKILLS_ASSESSMENT"
                ]
            }
        ],
        "description_l10n_id": "domain.education.description",
        "id": "EDUCATION",
        "name_l10n_id": "domain.education.name"
    },
    {
        "classes": [
            {
                "id": "CERTIFICATIONS_ELIGIBILITY",
                "name_l10n_id": "class.certifications_eligibility.name",
                "supported_purposes": [
                    "VACCINATION_STATUS_VERIFICATION",
                    "FITNESS_STATUS_VERIFICATION",
                    "HEALTH_RESTRICTED_ACCESS"
                ]
            },
            {
                "id": "MEDICAL_RECORDS",
                "name_l10n_id": "class.medical_records.name",
                "supported_purposes": [
                    "HEALTHCARE_SERVICES_ACCESS",
                    "MEDICAL_RECORDS_SHARING",
                    "MEDICAL_HISTORY_VALIDATION"
                ]
            }
        ],
        "description_l10n_id": "domain.health.description",
        "id": "HEALTH",
        "name_l10n_id": "domain.health.name"
    },
    {
        "classes": [
            {
                "id": "PAYMENT_INSTRUMENTS",
                "name_l10n_id": "class.payment_instruments.name",
                "supported_purposes": [
                    "PAYMENT_AUTHORIZATION",
                    "PAYMENT_EXECUTION",
                    "PROOF_OF_PAYMENT"
                ]
            },
            {
```

```
                "id": "PAYMENT_CREDENTIALS_AUTHORIZATIONS",
                "name_l10n_id": "class.payment_credentials_authorizations.name",
                "supported_purposes": [
                    "FINANCIAL_AUTHORIZATION_MANAGEMENT",
                    "STRONG_CUSTOMER_AUTHENTICATION"
                ]
            },
            {
                "id": "PUBLIC_PAYMENTS_FEES",
                "name_l10n_id": "class.public_payments_fees.name",
                "supported_purposes": [
                    "PROOF_OF_TAX_PAYMENT",
                    "PROOF_OF_FEE_PAYMENT",
                    "DIGITAL_STAMP_DUTY_VALIDATION"
                ]
            },
            {
                "id": "RECURRING_PAYMENTS_SUBSCRIPTIONS",
                "name_l10n_id": "class.recurring_payments_subscriptions.name",
                "supported_purposes": [
                    "RECURRING_PAYMENT_MANAGEMENT",
                    "SUBSCRIPTION_MANDATE_VERIFICATION"
                ]
            }
        ],
        "description_l10n_id": "domain.financial.description",
        "id": "FINANCIAL",
        "name_l10n_id": "domain.financial.name"
    },
    {
        "classes": [
            {
                "id": "CULTURAL_CARDS_BENEFITS",
                "name_l10n_id": "class.cultural_cards_benefits.name",
                "supported_purposes": [
                    "CULTURAL_SERVICES_ACCESS",
                    "LEISURE_SERVICES_ACCESS",
                    "MEMBER_DISCOUNT_APPLICATION"
                ]
            },
            {
                "id": "MEMBERSHIP_LOYALTY_PROGRAMS",
                "name_l10n_id": "class.membership_loyalty_programs.name",
                "supported_purposes": [
                    "AFFILIATION_VERIFICATION",
                    "PARTICIPATION_VERIFICATION",
                    "LOYALTY_BENEFITS_USE"
```

```
                ]
            }
        ],
        "description_l10n_id": "domain.culture_leisure.description",
        "id": "CULTURE_LEISURE",
        "name_l10n_id": "domain.culture_leisure.name"
    },
    {
        "classes": [
            {
                "id": "EMPLOYMENT_DOCUMENTS",
                "name_l10n_id": "class.employment_documents.name",
                "supported_purposes": [
                    "EMPLOYMENT_STATUS_VERIFICATION",
                    "PROFESSIONAL_PROFILE_VALIDATION"
                ]
            },
            {
                "id": "EMPLOYMENT_STATUS",
                "name_l10n_id": "class.employment_status.name",
                "supported_purposes": [
                    "CONTRIBUTION_RECORDS_VERIFICATION",
                    "EMPLOYMENT_BENEFITS_ELIGIBILITY"
                ]
            },
            {
                "id": "EMPLOYMENT_AFFILIATION",
                "name_l10n_id": "class.employment_affiliation.name",
                "supported_purposes": [
                    "ACCESS_PERMIT"
                ]
            }
        ],
        "description_l10n_id": "domain.employment.description",
        "id": "EMPLOYMENT",
        "name_l10n_id": "domain.employment.name"
    },
    {
        "classes": [
            {
                "id": "LICENSES_AUTHORIZATIONS",
                "name_l10n_id": "class.licenses_authorizations.name",
                "supported_purposes": [
                    "DRIVING_RIGHTS_VERIFICATION",
                    "NAVIGATION_RIGHTS_VERIFICATION",
                    "LAW_ENFORCEMENT_CONTROLS"
                ]
```

```
        },
        {
            "id": "VEHICLE_DOCUMENTS",
            "name_l10n_id": "class.vehicle_documents.name",
            "supported_purposes": [
                "VEHICLE_REGISTRATION_VERIFICATION",
                "VEHICLE_INSPECTION_VERIFICATION",
                "INSURANCE_STATUS_CHECK"
            ]
        },
        {
            "id": "TRANSPORT_SUBSCRIPTIONS",
            "name_l10n_id": "class.transport_subscriptions.name",
            "supported_purposes": [
                "TRANSPORT_SERVICES_ACCESS",
                "PUBLIC_TRANSPORT_PASS_VERIFICATION"
            ]
        },
        {
            "id": "TRAVEL_DOCUMENTS",
            "name_l10n_id": "class.travel_documents.name",
            "supported_purposes": [
                "RIGHT_TO_TRAVEL",
                "CROSS_BORDER_IDENTITY_CHECK"
            ]
        },
        {
            "id": "TRAVEL_INSURANCE_BOOKINGS",
            "name_l10n_id": "class.travel_insurance_bookings.name",
            "supported_purposes": [
                "TRAVEL_INSURANCE_VERIFICATION",
                "ACCOMMODATION_RESERVATION_CHECK",
                "TRANSPORT_RESERVATION_CHECK"
            ]
        },
        {
            "id": "DISCOUNTS_BENEFITS",
            "name_l10n_id": "class.discounts_benefits.name",
            "supported_purposes": [
                "MEMBER_DISCOUNT_APPLICATION",
                "TOURIST_BENEFITS_ACCESS"
            ]
        }
    ],
    "description_l10n_id": "domain.mobility_travel.description",
    "id": "MOBILITY_TRAVEL",
    "name_l10n_id": "domain.mobility_travel.name"
```

```
        },
        {
            "classes": [
                {
                    "id": "ECONOMIC_BENEFITS_ALLOWANCES",
                    "name_l10n_id": "class.economic_benefits_allowances.name",
                    "supported_purposes": [
                        "FAMILY_BENEFITS_ELIGIBILITY",
                        "UNEMPLOYMENT_BENEFITS_ELIGIBILITY",
                        "ECONOMIC_SUPPORT_ALLOCATION"
                    ]
                },
                {
                    "id": "INCENTIVES_VOUCHERS",
                    "name_l10n_id": "class.incentives_vouchers.name",
                    "supported_purposes": [
                        "DIGITAL_VOUCHER_USE",
                        "PURCHASE_INCENTIVE_USE",
                        "CASHBACK_ELIGIBILITY_VERIFICATION"
                    ]
                },
                {
                    "id": "HEALTH_WELLBEING_BONUSES",
                    "name_l10n_id": "class.health_wellbeing_bonuses.name",
                    "supported_purposes": [
                        "HEALTHCARE_BONUS_ACCESS",
                        "MENTAL_HEALTH_VOUCHER_USE",
                        "SPORTS_VOUCHER_USE"
                    ]
                }
            ],
            "description_l10n_id": "domain.bonuses.description",
            "id": "BONUSES",
            "name_l10n_id": "domain.bonuses.name"
        }
    ],
    "purposes": [
        {
            "id": "IDENTITY_VERIFICATION",
            "name_l10n_id": "purpose.identity_verification.name"
        },
        {
            "id": "AGE_VERIFICATION",
            "name_l10n_id": "purpose.age_verification.name"
        },
        {
            "id": "PERSON_IDENTIFICATION",
```

```
        "name_l10n_id": "purpose.person_identification.name"
    },
    {
        "id": "CIVIL_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.civil_status_verification.name"
    },
    {
        "id": "RIGHT_OF_RESIDENCE",
        "name_l10n_id": "purpose.right_of_residence.name"
    },
    {
        "id": "HOUSEHOLD_COMPOSITION_VERIFICATION",
        "name_l10n_id": "purpose.household_composition_verification.name"
    },
    {
        "id": "ELIGIBILITY_VERIFICATION",
        "name_l10n_id": "purpose.eligibility_verification.name"
    },
    {
        "id": "LEGAL_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.legal_status_verification.name"
    },
    {
        "id": "CRIMINAL_RECORD_CHECK",
        "name_l10n_id": "purpose.criminal_record_check.name"
    },
    {
        "id": "RESIDENCE_HOUSEHOLD_VERIFICATION",
        "name_l10n_id": "purpose.residence_household_verification.name"
    },
    {
        "id": "PROPERTY_OWNERSHIP_VERIFICATION",
        "name_l10n_id": "purpose.property_ownership_verification.name"
    },
    {
        "id": "REAL_ESTATE_COMPLIANCE",
        "name_l10n_id": "purpose.real_estate_compliance.name"
    },
    {
        "id": "FAMILY_SOCIAL_SERVICES_ELIGIBILITY",
        "name_l10n_id": "purpose.family_social_services_eligibility.name"
    },
    {
        "id": "LOCAL_TAX_COMPLIANCE",
        "name_l10n_id": "purpose.local_tax_compliance.name"
    },
    {
```

```
        "id": "PROPERTY_TAX_VERIFICATION",
        "name_l10n_id": "purpose.property_tax_verification.name"
    },
    {
        "id": "QUALIFICATION_DEGREE_VERIFICATION",
        "name_l10n_id": "purpose.qualification_degree_verification.name"
    },
    {
        "id": "EDUCATION_PATHWAY_ELIGIBILITY",
        "name_l10n_id": "purpose.education_pathway_eligibility.name"
    },
    {
        "id": "PROFESSIONAL_LICENSE_VERIFICATION",
        "name_l10n_id": "purpose.professional_license_verification.name"
    },
    {
        "id": "SKILLS_ASSESSMENT",
        "name_l10n_id": "purpose.skills_assessment.name"
    },
    {
        "id": "VACCINATION_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.vaccination_status_verification.name"
    },
    {
        "id": "FITNESS_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.fitness_status_verification.name"
    },
    {
        "id": "HEALTH_RESTRICTED_ACCESS",
        "name_l10n_id": "purpose.health_restricted_access.name"
    },
    {
        "id": "HEALTHCARE_SERVICES_ACCESS",
        "name_l10n_id": "purpose.healthcare_services_access.name"
    },
    {
        "id": "MEDICAL_RECORDS_SHARING",
        "name_l10n_id": "purpose.medical_records_sharing.name"
    },
    {
        "id": "MEDICAL_HISTORY_VALIDATION",
        "name_l10n_id": "purpose.medical_history_validation.name"
    },
    {
        "id": "PAYMENT_AUTHORIZATION",
        "name_l10n_id": "purpose.payment_authorization.name"
    },
```

```
{
    "id": "PAYMENT_EXECUTION",
    "name_l10n_id": "purpose.payment_execution.name"
},
{
    "id": "PROOF_OF_PAYMENT",
    "name_l10n_id": "purpose.proof_of_payment.name"
},
{
    "id": "FINANCIAL_AUTHORIZATION_MANAGEMENT",
    "name_l10n_id": "purpose.financial_authorization_management.name"
},
{
    "id": "STRONG_CUSTOMER_AUTHENTICATION",
    "name_l10n_id": "purpose.strong_customer_authentication.name"
},
{
    "id": "PROOF_OF_TAX_PAYMENT",
    "name_l10n_id": "purpose.proof_of_tax_payment.name"
},
{
    "id": "PROOF_OF_FEE_PAYMENT",
    "name_l10n_id": "purpose.proof_of_fee_payment.name"
},
{
    "id": "DIGITAL_STAMP_DUTY_VALIDATION",
    "name_l10n_id": "purpose.digital_stamp_duty_validation.name"
},
{
    "id": "RECURRING_PAYMENT_MANAGEMENT",
    "name_l10n_id": "purpose.recurring_payment_management.name"
},
{
    "id": "SUBSCRIPTION_MANDATE_VERIFICATION",
    "name_l10n_id": "purpose.subscription_mandate_verification.name"
},
{
    "id": "CULTURAL_SERVICES_ACCESS",
    "name_l10n_id": "purpose.cultural_services_access.name"
},
{
    "id": "LEISURE_SERVICES_ACCESS",
    "name_l10n_id": "purpose.leisure_services_access.name"
},
{
    "id": "MEMBER_DISCOUNT_APPLICATION",
    "name_l10n_id": "purpose.member_discount_application.name"
```

```
},
{
    "id": "AFFILIATION_VERIFICATION",
    "name_l10n_id": "purpose.affiliation_verification.name"
},
{
    "id": "PARTICIPATION_VERIFICATION",
    "name_l10n_id": "purpose.participation_verification.name"
},
{
    "id": "LOYALTY_BENEFITS_USE",
    "name_l10n_id": "purpose.loyalty_benefits_use.name"
},
{
    "id": "EMPLOYMENT_STATUS_VERIFICATION",
    "name_l10n_id": "purpose.employment_status_verification.name"
},
{
    "id": "PROFESSIONAL_PROFILE_VALIDATION",
    "name_l10n_id": "purpose.professional_profile_validation.name"
},
{
    "id": "CONTRIBUTION_RECORDS_VERIFICATION",
    "name_l10n_id": "purpose.contribution_records_verification.name"
},
{
    "id": "EMPLOYMENT_BENEFITS_ELIGIBILITY",
    "name_l10n_id": "purpose.employment_benefits_eligibility.name"
},
{
    "id": "DRIVING_RIGHTS_VERIFICATION",
    "name_l10n_id": "purpose.driving_rights_verification.name"
},
{
    "id": "NAVIGATION_RIGHTS_VERIFICATION",
    "name_l10n_id": "purpose.navigation_rights_verification.name"
},
{
    "id": "LAW_ENFORCEMENT_CONTROLS",
    "name_l10n_id": "purpose.law_enforcement_controls.name"
},
{
    "id": "VEHICLE_REGISTRATION_VERIFICATION",
    "name_l10n_id": "purpose.vehicle_registration_verification.name"
},
{
    "id": "VEHICLE_INSPECTION_VERIFICATION",
```

```
        "name_l10n_id": "purpose.vehicle_inspection_verification.name"
    },
    {
        "id": "INSURANCE_STATUS_CHECK",
        "name_l10n_id": "purpose.insurance_status_check.name"
    },
    {
        "id": "TRANSPORT_SERVICES_ACCESS",
        "name_l10n_id": "purpose.transport_services_access.name"
    },
    {
        "id": "PUBLIC_TRANSPORT_PASS_VERIFICATION",
        "name_l10n_id": "purpose.public_transport_pass_verification.name"
    },
    {
        "id": "RIGHT_TO_TRAVEL",
        "name_l10n_id": "purpose.right_to_travel.name"
    },
    {
        "id": "CROSS_BORDER_IDENTITY_CHECK",
        "name_l10n_id": "purpose.cross_border_identity_check.name"
    },
    {
        "id": "TRAVEL_INSURANCE_VERIFICATION",
        "name_l10n_id": "purpose.travel_insurance_verification.name"
    },
    {
        "id": "ACCOMMODATION_RESERVATION_CHECK",
        "name_l10n_id": "purpose.accommodation_reservation_check.name"
    },
    {
        "id": "TRANSPORT_RESERVATION_CHECK",
        "name_l10n_id": "purpose.transport_reservation_check.name"
    },
    {
        "id": "TOURIST_BENEFITS_ACCESS",
        "name_l10n_id": "purpose.tourist_benefits_access.name"
    },
    {
        "id": "FAMILY_BENEFITS_ELIGIBILITY",
        "name_l10n_id": "purpose.family_benefits_eligibility.name"
    },
    {
        "id": "UNEMPLOYMENT_BENEFITS_ELIGIBILITY",
        "name_l10n_id": "purpose.unemployment_benefits_eligibility.name"
    },
    {
```

```
            "id": "ECONOMIC_SUPPORT_ALLOCATION",
            "name_l10n_id": "purpose.economic_support_allocation.name"
        },
        {
            "id": "DIGITAL_VOUCHER_USE",
            "name_l10n_id": "purpose.digital_voucher_use.name"
        },
        {
            "id": "PURCHASE_INCENTIVE_USE",
            "name_l10n_id": "purpose.purchase_incentive_use.name"
        },
        {
            "id": "CASHBACK_ELIGIBILITY_VERIFICATION",
            "name_l10n_id": "purpose.cashback_eligibility_verification.name"
        },
        {
            "id": "HEALTHCARE_BONUS_ACCESS",
            "name_l10n_id": "purpose.healthcare_bonus_access.name"
        },
        {
            "id": "MENTAL_HEALTH_VOUCHER_USE",
            "name_l10n_id": "purpose.mental_health_voucher_use.name"
        },
        {
            "id": "SPORTS_VOUCHER_USE",
            "name_l10n_id": "purpose.sports_voucher_use.name"
        },
        {
            "id": "ACCESS_PERMIT",
            "name_l10n_id": "purpose.access_permit.name"
        }
    ]
}
```

Note

For a better and more efficient management of the localization of the Taxonomy, an Entity consulting it SHOULD:

- Download the basic version of the Taxonomy (compact, without localizations) using the `.well-known/taxonomy` endpoint.
- Determine the User's preferred language.
- Download only the necessary localization bundles.
- Dynamically merge localised content with the Taxonomy structure.

A non-normative example of a localization bundle output is given below:


```
{
  "taxonomy.name": "IT-Wallet Taxonomy",
  "taxonomy.description": "Hierarchical classification system for Digital Credentials in the IT
-Wallet ecosystem",
  "domain.identity.name": "Identity",
  "domain.identity.description": "Credentials that establish or confirm a person's legal identi
ty and personal, civil or legal status.",
  "class.identification_documents.name": "Identification Documents",
  "purpose.identity_verification.name": "Identity verification",
  "domain.authentication.name": "Authentication",
  "domain.authentication.description": "Credentials that attest authorisation to access restric
ted physical or digital spaces, services or resources.",
  "class.access.name": "Access",
  "purpose.access_permit.name": "Access permit verification",
  "...": "..."
}
```


Localization bundles MUST be available at the URI composed by appending the locale code and `.json` to the `localization.base_uri` value defined in the taxonomy. Each locale bundle MUST be accessible following the naming pattern **{locale_code}.json**, where **{locale_code}** is replaced with the corresponding locale code from the **available_locales** array.

A non-normative example of the Italian localization URI for the bundle would be **https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/it.json**.

7.8 Schema Registry

The **Schema Registry** is the authoritative inventory of all known and accepted **Credential Schemas** (JSON Schema for SD-JWT, CBOR Schema for mDOC) within the IT-Wallet ecosystem. It is managed by the Trust Anchor and provides a single, verifiable source for retrieving the technical specifications required for parsing, validating, and displaying Digital Credentials.

Schema Registry Objectives:

1. **Schema Centralization**: Provide a centralized access point for all technical schemata used by Digital Credentials.
2. **Integrity and Authenticity**: Ensure the integrity and authenticity of the schema documents through cryptographic digests.
3. **Interoperability**: Facilitate the seamless integration of Wallet Providers and Relying Parties by providing consistent schema versions.
4. **Credential Lifecycle Support**: Act as a verifiable reference point for schema validation during issuance and presentation.

Schema Registry Structure and Access:

The Schema Registry is accessible via the `.well-known/it-wallet-registry` discovery endpoint under the *schema_registry* field. It allows for the discovery of schema URIs and their cryptographic integrity checks.

Table 7.13: First-level Fields of the Schema Registry

Field Name	Description
id	REQUIRED. Unique identifier of the Schema Registry (e.g., `urn:schemas:it-wallet`).
version	REQUIRED. The version of the Schema Registry (e.g., `1.0.0`).
last_modified	REQUIRED. The timestamp indicating when the Schema Registry was last updated (e.g., `2025-03-15T12:00:00Z`).
schemas	REQUIRED. A JSON Array where each entry is a JSON Object representing a Credential Schema definition. Each object contains the parameters defined in the "Schema Definition Parameters" table below, including schema identification, format specifications, URIs, and integrity verification data.

Table 7.14: Schema Definition Parameters

Field Name	Description
id	REQUIRED. The unique identifier of the scheme (e.g., `mDL+mso_mdoc+org.iso.18013.5.1.mDL`).
version	REQUIRED. The version of the schema definition (e.g., `1.0.0`).
credential_type	REQUIRED. The unique identifier of the Digital Credential type (e.g., `mDL`, `pid`, `eid`).
format	REQUIRED. The technical format of the schema (e.g., `mso_mdoc`, `dc+sd-jwt`).
vct	CONDITIONAL. It is REQUIRED if the `format` is `dc+sd-jwt`, indicating the Verifiable Credential Type (e.g., `urn:eudi:mDL:it:1`).
docType	CONDITIONAL. It is REQUIRED if the `format` is `mso_mdoc`, indicating the document type used (e.g., `org.iso.18013.5.1.mDL`).
schema_uri	REQUIRED. The URI where the schema document can be retrieved (e.g. , `https://trust-registry.it-wallet.example.it/.well-known/schemas/mdoc/mDL`).
schema_uri#integrity	REQUIRED. Cryptographic digest of the schema document for integrity verification. Format: `{digest_method}-{digest_value}` (e.g., `sha256-c8b708728e4c5756e35c03aeac257ca878d1f717d7b61f621be4d36dbd9b9c16`).
description	OPTIONAL. A human-readable description of the schema, which may be localized (e.g., "Schema tecnico per la mobile Driving License in formato mdoc.").

Schema Registry Example:

A non-normative example of the Schema Registry payload:


```
{
    "id": "urn:schemas:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "schemas": [
        {
            "id": "mDL+mso_mdoc+org.iso.18013.5.1.mDL",
            "version": "1.0.0",
            "credential_type": "mDL",
            "format": "mso_mdoc",
            "docType": "org.iso.18013.5.1.mDL",
            "schema_uri": "https://trust-registry.it-wallet.example.it/.well-known/schemas/mdoc
/mDL",
            "schema_uri#integrity": "sha256-c8b708728e4c5756e35c03aeac257ca878d1f717d7b61f621be
4d36dbd9b9c16",
            "description": "Schema tecnico per la mobile Driving License in formato mdoc."
        },
        {
            "id": "pid+dc+sd-jwt+urn:eudi:pid:it:1",
            "version": "1.0.0",
            "credential_type": "pid",
            "format": "dc+sd-jwt",
            "vct": "urn:eudi:pid:it:1",
            "schema_uri": "https://trust-registry.it-wallet.example.it/.well-known/schemas/sd-j
wt/pid",
            "schema_uri#integrity": "sha256-a1b2c3d4e5f67890...",
            "description": "Schema tecnico per la Person Identification Data (PID) in formato S
D-JWT."
        }
    ]
}
```


7.9 Registry Integration and Cross-References

The registry components are interconnected and work together to support the complete Credential ecosystem:

1. **AS Registry ↔ Taxonomy**: AS entities declare capabilities using taxonomy classifications for standardized categorization.
2. **AS Registry ↔ Catalog**: Credential types reference AS capabilities for data source validation.

3. **Catalog ↔ Taxonomy**: Credential entries specify domains and purposes from the taxonomy for discovery and authorization.
4. **Federation Registry ↔ All Components**: Provides cryptographic trust validation for all registry operations and entity authentication.
5. **Schema Registry ↔ Issuer/RPs**: Provides the verifiable link to all known Credential format specifications used in the ecosystem.

7.10 Registry Infrastructure Usage Journeys

The components of the Registry Infrastructure are designed to support various operational phases of the IT-Wallet ecosystem, each involving specific interactions between entities. The main Journeys below illustrate the interactions with the Registry Infrastructure.

7.10.1 Catalog Browsing

This *Catalog Browsing* journey supports Users (both human users via a **Wallet Instance** and automated systems like **Relying Parties** or web portals) in discovering and selecting available Digital Credentials.

1. **Accessing the Discovery Endpoint**: The entity (e.g., a Wallet Provider or informational portal) accesses the *Registry Discovery Endpoint* (`.well-known/it-wallet-registry`) to obtain the URI of the **Digital Credentials Catalog** and of the **Taxonomy**.
2. **Navigation and Selection**:

- **Credential Discovery**: The entity browses the list of Credentials (`credentials` field) to identify relevant Credential types (e.g., `pid`, `eid`, `mDL`) and, if needed, uses the information on the **Taxonomy** to navigate their hierarchy and to provide different localizations.
- **Issuer Metadata**: The entity extracts the **Issuer Identifier** (*entity_id* within the *issuers* field) associated with the desired Credential.
- **Detail Consultation**: To obtain complete information oand specific technical requirements, the entity accesses the **Entity Configuration** (Issuer Metadata) using the retrieved identifier.

3. **Final Action**: The entity can then can use the metadata to display the catalog information to a User (or use the information in other way).

7.10.2 Credential Issuance

This journey defines how a Credential Issuer uses the Registry Infrastructure to prepare and issue a compliant Digital Credential.

1. **Identifying Requirements**: The CI consults the **Digital Credentials Catalog** for the technical requirements of the Credential type to be issued (e.g., `max_validity_days`, `min_loa`).
2. **Schema and Claim Resolution**:

- The CI consults the **Schema Registry** to retrieve the technical specification of the format and schema (e.g., JSON Schema for SD-JWT) required by the Catalog, ensuring validity and integrity via the hash (*schema_uri#integrity*).
- The CI accesses the **Claims Registry** to retrieve the standardized semantic definitions and data formats (data types) of the necessary attributes (claims).

3. **Authentic Data Retrieval**:

- The CI consults the **Authentic Source (AS) Registry** to identify the authorized **Authentic Source** (AS) for the required dataset. The AS Registry provides the AS's `entity_id` and the technical details of the interface (*integration_endpoint*, *integration_method*).
- The CI consults the AS endpoint specification to implement the integration needed to retrieve the User data required to populate the Digital Credential.

4. **Credential Issuance**: The CI uses the retrieved data, validated schemas, and specified formats to generate and sign the Digital Credential in the correct format (e.g., SD-JWT or mDOC).

7.10.3 Credential Presentation and Verification

This journey describes how a **Wallet Instance** and a **Relying Party (RP)** interact with the Registry Infrastructure when a Digital Credential needs to be presented by a User.

1. **Wallet Authorization and Selection**:

- The Wallet receives a Presentation Request from the RP, verifies the validity of the request comparing the requested *claims* with the *Authorization Policies* related to the RP .
- The Wallet consults the **Digital Credentials Catalog** and the **Taxonomy** to verify the *Domains*, the *Classes* and *Purposes* associated with the Credential types it holds, evaluating which Credentials are suitable for the request.
- The Wallet verifies if the required attributes (claims) are available and authorized for disclosure based on the request policy (**Credential-Specific** or **Credential-Agnostic** scenarios).

- The User authorizes the release of the selected, selectively disclosed attributes. The Wallet then packages and presents the Digital Credential to the RP.

2. **Discovery and Integrity**:

- The RP receives the Digital Credential from the User.
- The RP consults the **Federation Registry** via the Trust Anchor's endpoint (*federation_resolve*, *federation_trust_mark_status*) to verify the **cryptographic trust** (Trust Mark) of the Issuer and Wallet Provider as defined in Section *The Infrastructure of Trust* (page 109).
- The RP consults the **Schema Registry** to download the schema of the presented Credential (*schema_uri*), verifying its integrity (*schema_uri#integrity*).

3. **Schema and Final Policy Validation**:

- The RP uses the retrieved schema to validate the structure of the Credential and the data types of the revealed attributes.
- The RP performs the final check to ensure that the attributes presented comply with the specific requirements of the initial request and authorization policy.

4. **Acceptance or Rejection**: Based on cryptographic validation, schema compliance, and policy-based authorization, the RP accepts or rejects the Credential for service access.

CHAPTER

EIGHT

ENTITY ONBOARDING

This section defines the technical specifications for entity lifecycle management in the IT-Wallet ecosystem based on the **Registry Infrastructure** defined in *Registry Infrastructure* (page 149). This includes initial onboarding procedures, ongoing management operations, such as data updates and modifications, and federation exit processes.

For a high-level overview of the onboarding process, see *Onboarding System* (page 23). The Section *Onboarding Journey Maps* (page 29) provides an onboarding journey map from the perspective of Entity's operators.

8.1 Overview

8.1.1 Entity Onboarding System Architecture

The IT-Wallet ecosystem operates on a federated trust infrastructure, requiring participating entities to establish mutual trust before engaging in any interactions involving User attributes.

The onboarding framework defines technical registration procedures based on the type of participant:

1. Authentic Sources follow data-focused registration processes.
2. Operational Entities (Credential Issuers, Relying Parties, Wallet Providers) follow trust establishment procedures.

8.1.2 Entity Types and Onboarding Pathways

The following table summarizes entity types, their roles, and corresponding onboarding pathways:

Table 8.1: Entity Types and Onboarding Pathways

Entity Type	Primary Role	Onboarding Pathway	Key Requirements
Authentic Sources	Authoritative data providers for Digital Credential attributes	*Authentic Sources Registration Process* (page 227)	Data authority validation; API integration (PDND)
Credential Issuers	Generate and issue Digital Credentials using Authentic Source data	*Federation Entities Onboarding Process* (page 241)	IT-Wallet Trust Infrastructure compliance; see *The Infrastructure of Trust* (page 109)
Relying Parties	Verify Digital Credentials for service access	*Federation Entities Onboarding Process* (page 241)	IT-Wallet Trust Infrastructure compliance; see *The Infrastructure of Trust* (page 109)
Wallet Providers	Provide wallet solutions to citizens	*Federation Entities Onboarding Process* (page 241)	IT-Wallet Trust Infrastructure compliance (see *The Infrastructure of Trust* (page 109)); Wallet Attestation capabilities (see *Wallet Instance Initialization and Registration* (page 285))
Wallet Instances	User-level digital wallet applications	Indirect registration via Wallet Provider, see *Wallet Instance Initialization and Registration* (page 285)	Wallet Attestation from trustworthy Wallet Provider

8.1.3 Administrative and Technical Registration

The onboarding process follows a structured multi-phase approach:

1. **Administrative Registration**: All entities MUST complete initial administrative registration that validates their legal standing, regulatory compliance, and organizational eligibility to participate in the IT-Wallet ecosystem.
2. **Technical Registration**: Following administrative approval, entities make technical registration through specialised pathways:

- **Authentic Source Registration**: Data-focused registration procedures with API integration validation.
- **Federation Registration**: Cryptographic trust establishment as defined in Section *The Infrastructure of Trust* (page 109).

3. **IT-Wallet Registry Integration**:

- **Claims Registry Integration**: Authentic Sources select standardized claim definitions from Claims Registry during capability declaration.
- **Taxonomy Integration**: All entities use Taxonomy hierarchical classification (domains, classes, purposes) for organizational structure to categorize Credentials.
- **Authentic Source Registry Integration**: Authentic Sources registered with their declared claims and capabilities, enabling Credential Issuers discovery and coordination.
- **Federation Registry Integration**: Operational entities included for trust validation during Credential operations.
- **Catalog Integration**: Credential types published in *Digital Credentials Catalog Structure* (page 194) based on supervisory body policies for public discovery eligibility.

All registry components and their interactions are detailed in *Registry Infrastructure* (page 149).

8.2 Authentic Sources Registration Process

Authentic Sources MUST complete a structured registration process to confirm their status as authoritative data providers in the IT-Wallet ecosystem. This process includes specifying requirements and undergoing procedural validation, as detailed in *Authentic Source Operator Journey* (page 31).

8.2.1 Authentic Source Registration Requirements

Authentic Sources MUST comply with the following technical requirements:

- **Claims Compliance**:
  - **Claims Registry Adoption**: The Entities MUST use standardized Claims Registry identifiers in data responses. Custom claim mapping is not allowed.
- **API Integration Standards**:
  - **Public Entities**: MUST integrate through PDND platform with e-service implementation following national specifications.
  - **Private Entities**: MUST provide a complete OAS3 Specification document that includes authorization framework, request/response schemas, error handling mechanisms, and sandbox environment for testing.
- **Response Format Standardization**:

  - **Standard Claims Format**: The Entities MUST use Claims Registry identifiers and formats in all data responses.
  - **State Mapping**: The Entities MUST handle clear mapping between their internal states and standard Credential states (valid, suspended, revoked).

- **Security and Quality Assurance**:
  - **Security Standards**: The Entities MUST implement TLS 1.3 or higher with robust authentication mechanisms, forward secrecy, and cryptographic algorithms that meet current and emerging security standards, end-to-end confidentiality and integrity of all communications, maintaining compliance with evolving regulatory requirements and industry best practices.
  - **User Authentication Evidence**: The Entities MAY request User authentication evidence from Credential Issuer before granting access to e-services for obtaining User attributes.
  - **Data Quality**: The Entities MUST specify update frequency and provide data freshness guarantees.
  - **Audit Trail**: The Entities MUST implement logging capabilities for all data access and provisioning activities.

8.2.2 Authentic Sources Registration Information Requirements

Note

Only Italian Authentic Sources can be onboarded at the current stage of IT-Wallet.

During registration, Authentic Sources MUST provide the following information:

Table 8.2: Authentic Source Registration Information Requirements

Information Category	Description and Examples
Organization Information	**REQUIRED**. Organization Details including: • Organization name, type ("public" or "private") and country (ISO 3166-1 alpha-2). • Administrative identifier codes such as IPA registration code (REQUIRED only for public Authentic Sources) and legal identifier (tax payer's number or VAT number). • Contact Information including technical and administrative contacts, such as email addresses, homepage URI, privacy policy URI.
Data Capabilities Declaration	**REQUIRED**. Available claims: • Array of claim identifiers from Claims Registry that the Authentic Source provides (e.g., `["given_name", "family_name", "driving_privileges"]`). • Intended Purposes for verification, using taxonomy purpose identifiers (e.g. `["DRIVING_RIGHTS_VERIFICATION"]`).
API Implementation Details	**REQUIRED**. Integration information details: • Authorization framework for API access. • API definitions such as Request/Response Formats, including error management.
Data Provision Capabilities	**REQUIRED**. Indicates if Authentic Source supports immediate/deferred data provision (boolean).
User Information	**OPTIONAL**. Markdown-formatted text containing human-readable information about data availability constraints or limitations. For example, if the Authentic Source database only contains data registered after a specific date, this information MUST be conveyed to Users. **Example**: "Driving license data is available for licenses issued after January 1, 2020. For older licenses, contact the local motorization office.".

continues on next page

Table 8.2 – continued from previous page

Information Category	Description and Examples
Display Properties	**OPTIONAL**. Visual branding suggestions for Digital Credentials using Authentic Source data: • Background color for Digital Credentials in hexadecimal format (e.g., `"#003d82"`). • Logo URI with cryptographic integrity verification for Digital Credential branding. • Visual template URI with integrity verification for Digital Credential presentation.

8.2.3 Authentic Source Registration Procedure

The Authentic Source registration follows a technical process as described below.

Step 1 - Registration Package Preparation: The Entity prepares registration information according to the requirements table above. A non-normative example of information in JSON format is provided below.

Table 8.3: First-level Fields of the Authentic Source Registration

Field Name	Description
id	REQUIRED. Unique identifier of the Authentic Source Registry (e.g., `urn:authentic-sources:it-wallet`).
version	REQUIRED. The version of the Authentic Source Registry (e.g., `1.0.0`).
last_modified	REQUIRED. The timestamp indicating when the list was last updated (e.g., `2025-03-15T12:00:00Z`).
localization	REQUIRED. Localization configuration object containing: • **default_locale**: Default locale code (e.g., `it`). • **available_locales**: Array of supported locale codes (e.g., `["en", "it"]`). • **base_uri**: Base URI for localization bundle retrieval (e.g., `https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/`). • **version**: Version of the localization bundle format.

continues on next page

Table 8.3 – continued from previous page

Field Name	Description
authentic_sources	REQUIRED. A JSON Array where each entry is a JSON Object representing an Authentic Source entity. Each object contains the parameters defined in the "Authentic Sources Parameters" table below, including entity identification, organizational information, data capabilities, and integration methods.

Table 8.4: Authentic Sources Parameters

Parameter	Type	Description
entity_id	string	REQUIRED. Unique identifier following the normative schema: `https://{organization_domain}[/{optional_path}]`.
organization_info	JSON object	REQUIRED. Legal entity details and organizational metadata.
organization_info. organization_name	JSON Object Array	REQUIRED. Object array containing the organization name in multiple languages. It MUST contain `locale` and `name` claims.
organization_info. organization_type	string	REQUIRED. Entity classification: `"public"` or `"private"`.
organization_info. ipa_code	string	REQUIRED only for Public AS. IPA registration code for government entities.
organization_info. legal_identifier	string	REQUIRED. Legal registration identifier (Fiscal Code/VAT Number, or equivalent national identifier for foreign entities).
organization_info. homepage_uri	string	REQUIRED. URL pointing to the organization's homepage.
organization_info. contacts	String Array	REQUIRED. Array of contact email addresses for at least one user-support, one application, and one systems specialist.
organization_info. dpa_contact	string	REQUIRED. An e-mail address of Authentic Source DPA.
organization_info. policy_uri	string	REQUIRED. URL to privacy policy document.
organization_info. tos_uri	string	OPTIONAL. URL to terms of service document.
organization_info. organization_country	string	REQUIRED. Two-letter ISO 3166-1 alpha-2 country code of the organization.
organization_info. logo_uri	string	OPTIONAL. URL to the organization's logo image.
organization_info. logo_uri#integrity	string	CONDITIONAL. Cryptographic digest of the logo image resource for integrity verification. REQUIRED if `logo_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).

continues on next page

Table 8.4 – continued from previous page

Parameter	Type	Description
organization_info.logo_alt_text_l10n_id	string	OPTIONAL. Alternative text for the organization's logo image.
organization_info.logo_extended_uri	string	OPTIONAL. URL to the organization's extended logo image.
organization_info.logo_extended_uri#integrity	string	CONDITIONAL. Cryptographic digest of the extended logo image resource for integrity verification. REQUIRED if `logo_extended_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).
organization_info.logo_extended_alt_text_l10n_id	string	OPTIONAL. Alternative text for the organization's extended logo image.
data_capabilities	JSON Objects Array	REQUIRED. Array containing data capability specifications.
data_capabilities[].dataset_id	string	REQUIRED. The *Dataset_id* within the scope of the Authentic Source, which MAY be used as a query parameter for the `GetAttributeClaims` service.
data_capabilities[].data_origin	JSON Object Array	REQUIRED. Object array containing the human-readable name of the data origin or department providing the data in multiple languages. It MUST contain `locale` and `name` claims.
data_capabilities[].intended_purposes	String Array	REQUIRED. Business purposes served, using taxonomy purpose identifiers (e.g., `["IDENTITY_VERIFICATION", "DRIVING_RIGHTS_VERIFICATION"]`).
data_capabilities[].available_claims	String Array	REQUIRED. Claims available from this data capability.
data_capabilities[].available_claims.claim_name	string	REQUIRED. It Contains the name of the claim.
data_capabilities[].available_claims.order	number	REQUIRED. Defines the order in which the information would be shown.
data_capabilities[].available_claims.mandatory	boolean	REQUIRED. Defines if a claim is always available or not.
data_capabilities[].integration_method	string	REQUIRED. Authorization framework used for data access. MUST be `"pdnd"`.
data_capabilities[].integration_endpoint	string	OPTIONAL. Service access point (PDND endpoint).
data_capabilities[].api_specification	string	OPTIONAL. URL to OAS3 specification document for this data capability.
data_capabilities[].data_provision	JSON object	OPTIONAL. Data provision capabilities and timing specifications.

continues on next page

Table 8.4 – continued from previous page

Parameter	Type	Description
data_capabilities[]. data_provision. immediate_flow	boolean	REQUIRED. Indicates if the Authentic Source supports immediate data provision.
data_capabilities[]. data_provision. deferred_flow	boolean	REQUIRED. Indicates if the Authentic Source supports deferred data provision.
data_capabilities[]. data_provision. max_response_ time_minutes	integer	CONDITIONAL. Maximum time in minutes for the Authentic Source to respond to a deferred data provision request. REQUIRED if `deferred_flow` is `true`.
data_capabilities[]. data_provision.notification_methods	String Array	CONDITIONAL. Array of notification methods supported by the Authentic Source for deferred data provision, such as `"push"`, `"poll"`. REQUIRED if `deferred_flow` is `true`.
data_capabilities[]. user_information	JSON object Array	OPTIONAL. Object array containing information about the data capability relevant to the User in multiple languages. This string MUST be provided by the Authentic Source to the Trust Anchor during onboarding. The Markdown formatting can be plain text or a combination of text and links. For example, if the Authentic Source's database only contains data registered *after* a specific date, this information MUST be conveyed through this key. It MUST contain `locale` and `description` claims.
data_capabilities[]. administrative_ expiration_user_ info	JSON object Array	OPTIONAL. Object array containing information in multiple languages for the User about the administrative expiration status of the dataset and, eventually recommended actions. This string MUST be provided by the Authentic Source to the Trust Anchor during onboarding. The Markdown formatting can be plain text or a combination of text and links. It MUST contain `locale`, `title` and `description` claims.
data_capabilities[]. allowed_states	JSON object Array	OPTIONAL. Object array containing information in multiple languages for the User about the current status of the dataset and, eventually recommended actions. The status values are defined in the *Token Status Lists* (page 411) Section. It MUST contain `locale`, `title`, `description` and `<Status-Type-Value>` claims.
data_capabilities[]. service_documentation_uri	string	OPTIONAL. URL pointing to the Authentic Source service documentation.
data_capabilities[]. update_frequency	string	OPTIONAL. Indicates how frequently the Authentic Source updates its data. Possible values: `"real_time"` (near real-time updates, typically within minutes), `"daily"`, `"weekly"`, `"monthly"`, `"on_demand"`.

continues on next page

Table 8.4 – continued from previous page

Parameter	Type	Description
data__capabilities[]. logo__uri	string	OPTIONAL. URL to the logo image related to the data.
data__capabilities[]. logo__uri#integrity	string	CONDITIONAL. Cryptographic digest of the logo image resource for integrity verification. REQUIRED if `logo_uri` is present. Format: `{digest_method}-{digest_value}` (e.g., `"sha-256-abc123..."`).
data__capabilities[]. background__color	string	OPTIONAL. String value of the background color related to be displayed together with the data.
data__capabilities[]. contacts	String Array	OPTIONAL. Array of customer service contact email addresses.


```
{
  "data_capabilities": [
    {
      "api_specification": "https://docs.gov.example/transport/api-oas3.yaml",
      "available_claims": [
        {
          "claim_name": "given_name",
          "mandatory": true,
          "order": 1
        },
        {
          "claim_name": "family_name",
          "mandatory": true,
          "order": 2
        },
        {
          "claim_name": "birth_date",
          "mandatory": true,
          "order": 3
        },
        {
          "claim_name": "birth_place",
          "mandatory": true,
          "order": 4
        },
        {
          "claim_name": "issue_date",
          "mandatory": true,
          "order": 5
        },
        {
          "claim_name": "document_iss_country",
          "mandatory": true,
```

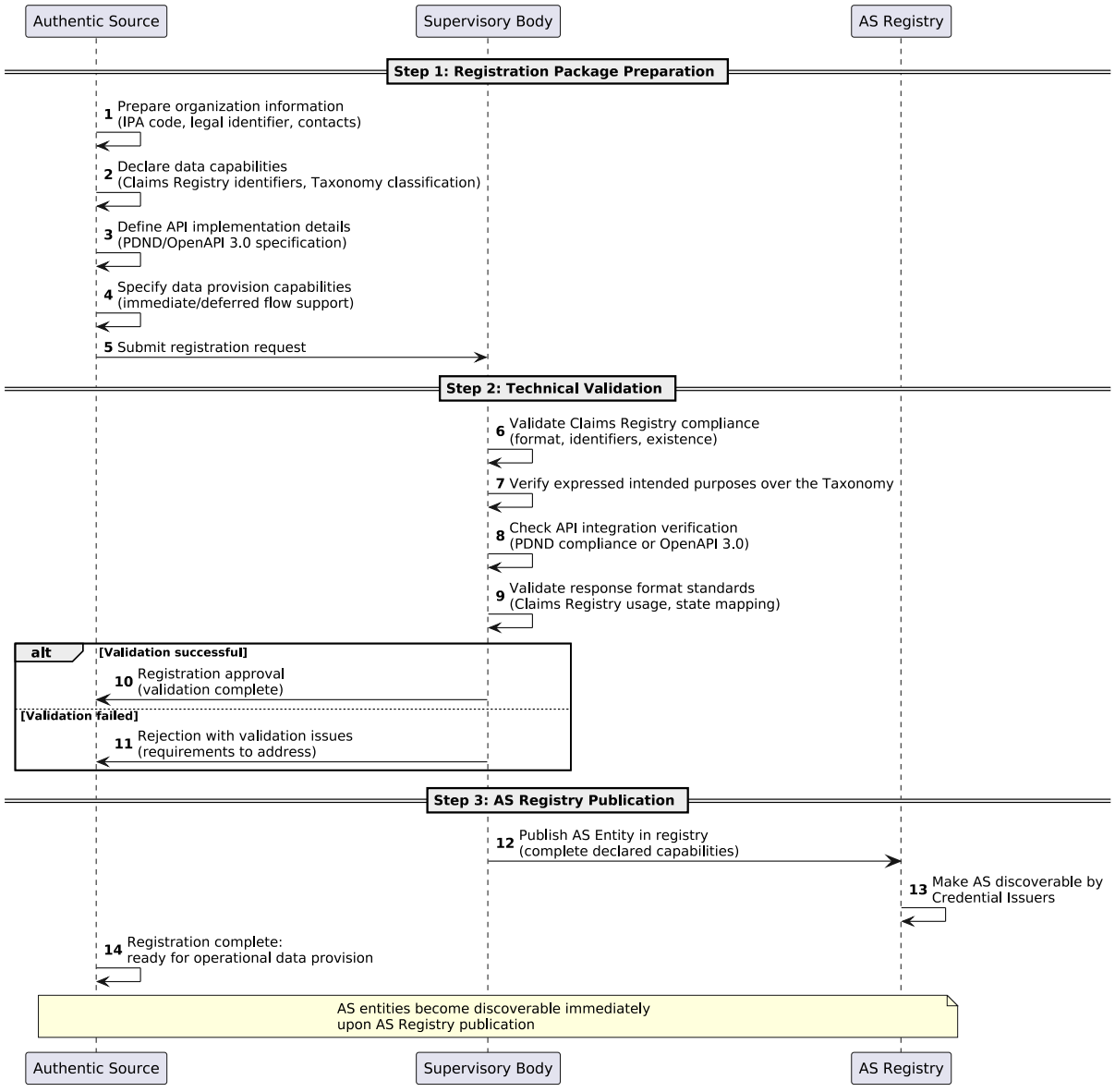


Fig. 8.1: Authentic Source Registration Process.

```
        "order": 6
      },
      {
        "claim_name": "document_iss_authority",
        "mandatory": true,
        "order": 7
      },
      {
        "claim_name": "expiry_date",
        "mandatory": true,
        "order": 8
      },
      {
        "claim_name": "document_number",
        "mandatory": true,
        "order": 9
      },
      {
        "claim_name": "portrait",
        "mandatory": true,
        "order": 10
      },
      {
        "claim_name": "un_distinguishing_sign",
        "mandatory": true,
        "order": 11
      },
      {
        "claim_name": "restrictions_conditions",
        "mandatory": true,
        "order": 12
      },
      {
        "claim_name": "driving_privileges",
        "mandatory": true,
        "order": 13
      }
    ],
    "administrative_expiration_user_info": [
      {
      "locale": "it-IT",
      "title": "La tua Patente risulta scaduta",
      "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Rinnova il documento presso l'ente abilitato del tuo territori."
      },
      {
      "locale": "en-US",
```

```
        "title": "Your driving licence has expired",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Renew the document at the authorised office in your area."
        }
      ],
      "allowed_states": [
        {
        "locale": "it-IT",
        "0x00": "VALID",
        "title": "La tua Patente risulta valida",
        "description": "In questo momento puoi guidare le categorie di veicoli associate alla t
ua Patente"
        },
        {
        "locale": "en-US",
        "0x00": "VALID",
        "title": "Your driving licence is valid",
        "description": "At this time, you can drive the vehicle categories associated with your
licence."
        },
        {
        "locale": "it-IT",
        "0x01": "INVALID",
        "title": "La tua Patente risulta non valida",
        "description": "Rivolgiti all'ente preposto per maggiori dettagli."
        },
        {
        "locale": "en-US",
        "0x01": "INVALID",
        "title": "Your driving licence is not valid",
        "description": "Contact the competent authority for more details."
        },
        {
        "locale": "it-IT",
        "0x02": "SUSPENDED",
        "title": "La tua Patente risulta sospesa",
        "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Rivolgiti all'ente preposto per maggiori dettagli sul termine del periodo di so
spensione."
        },
        {
        "locale": "en-US",
        "0x02": "SUSPENDED",
        "title": "Your driving licence is suspended",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Contact the competent authority for more details on the end of the suspension peri
od."
```

```
        },
        {
        "locale": "it-IT",
        "0x0F": "ATTRIBUTE_UPDATE",
        "title": "La tua Patente risulta essere stata aggiornata",
        "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Aggiorna il documento su IT-Wallet."
        },
        {
        "locale": "en-US",
        "0x0F": "ATTRIBUTE_UPDATE",
        "title": "Your driving licence has been updated",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Update the document in the IT-Wallet."
        }
      ],
      "background_color": "#12107c",
      "contacts": [
        "citizen-support@gov.example"
      ],
      "data_origin": [
        {
        "locale": "it-IT",
        "name":"Ufficio/Dipartimento XXX Autorità dei Trasporti"
        },
        {
        "locale": "en-US",
        "name":"Office/Department XXX Transport Authority"
        }
      ],
      "data_provision": {
        "deferred_flow": false,
        "immediate_flow": true
      },
      "dataset_id": "38832801",
      "integration_endpoint": "https://api.gov.example/transport/driving-license",
      "integration_method": "pdnd",
      "intended_purposes": [
        "IDENTITY_VERIFICATION",
        "PERSON_IDENTIFICATION",
        "DRIVING_RIGHTS_VERIFICATION",
        "LAW_ENFORCEMENT_CONTROLS"
      ],
      "logo_uri": "https://www.gov.example/assets/transport-data-logo.svg",
      "logo_uri#integrity": "sha-256-b1946ac92492d2347c6235b4d2611184e2a3f8b8c8f8f8e8f8f8f8f8f8
f8f8f8",
      "service_documentation_uri": "https://docs.gov.example/transport/api-docs",
```

```
      "update_frequency": "real_time",
      "user_information": [
        {
        "locale": "it-IT",
        "description": "Sono disponibili le patenti di guida rilasciate o rinnovate a partire d
al DD-MM-YYYY."
        },
        {
        "locale": "en-US",
        "description": "Only driving licences issued or renewed from DD-MM-YYYY will be availab
le."
        }
      ]
    }
  ],
  "entity_id": "https://transport-authority.gov.example",
  "organization_info": {
    "contacts": [
      "registry@transport-authority.gov.example",
      "technical-support@transport-authority.gov.example"
    ],
    "homepage_uri": "https://www.gov.example/transport",
    "ipa_code": "nta_001",
    "legal_identifier": "12345678901",
    "logo_extended_uri": "https://www.gov.example/assets/transport-authority-logo-extended.svg"
,
    "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998
e86f7f7a27ae3",
    "logo_uri": "https://www.gov.example/assets/transport-authority-logo.svg",
    "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a2
7ae3",
    "organization_country": "XX",
    "organization_name": [
      {
      "locale": "it-IT",
      "name": "Autorità dei Trasporti"
      },
      {
      "locale": "en-US",
      "name": "Transport Authority"
      }
    ],

    "organization_type": "public",
    "policy_uri": "https://www.gov.example/transport/privacy-policy",
    "tos_uri": "https://www.gov.example/transport/tos"
```

```
  }
}
```

Step 2 - Technical Validation: Supervisory Body validates submitted registration focusing on:

- **Claims Registry Compliance**: Validation of claims format, identifiers, and existence in Claims Registry.
- **Taxonomy Validation**: Verification that declared purposes are valid taxonomy entries.
- **API Integration Verification**:
  - **Public Entities**: e-Service PDNDspecification compliance verification
  - **Private Entities**: OAS3 specification completeness including authorization framework, request/response schemas, error handling mechanisms, and sandbox environment.
- **Response Format Standards**: Verification of Claims Registry format usage and state mapping specification.

Step 3 - Authentic Source Registry Publication: Upon successful validation:

- Authentic Source Entity is published in the Authentic Source Registry with complete declared capabilities.
- Authentic Source becomes discoverable by Credential Issuers for integration requests.
- Authentic Source is ready for operational data provision.

Note

Authentic Source registration is complete and independent of Credential Issuer integration. Authentic Sources become discoverable immediately upon Authentic Source Registry publication, while Credential availability to end Users depends on administrative Authentic Source to Credential Issuer authorization followed by successful technical integration and Supervisory Body policy approval for catalog eligibility.

8.3 Authentic Source to Credential Issuer Authorization Process

Following administrative Authentic Source to Credential Issuer authorization obtained during the administrative registration phase, technical integration procedures establish the operational API connections and data access mechanisms between Credential Issuers and Authentic Sources.

Technical integration encompasses:

- **API Endpoint Configuration**: Establishment of secure API connections as specified in Authentic Source technical specifications (PDND e-services).
- **Claims Mapping Validation**: Verification that Credential Issuer implementation correctly maps Authentic Source data responses to standardized Claims Registry identifiers.
- **Data Flow Testing**: Validation of immediate or deferred data provision capabilities and error handling mechanisms.
- **Security Implementation**: Configuration of authentication, authorization, and audit logging as required by Authentic Source security standards.

8.4 Federation Entities Onboarding Process

Federation Entities, including Credential Issuers, Relying Parties, and Wallet Providers, must complete onboarding procedures to become eligible participants in the IT-Wallet ecosystem. This process establishes distributed trust by issuing X.509 Certificates, validating Trust Chains, and confirming compliance, as described in *The Infrastructure of Trust* (page 109).

8.4.1 Hierarchical Federation Authority Model

The IT-Wallet federation implements a **hierarchical onboarding model** where Federation Entities MUST be onboarded by one of the following actors:

1. **Trust Anchor**: The root authority that has the capability to directly onboard any Federation Entity.
2. **Intermediates**: Delegated authorities that onboard Leaf Entities on behalf of Trust Anchor.

Both Trust Anchors and Intermediates act as **Federation Authorities** with the following onboarding capabilities:

- **X.509 Certificate Issuance**: Issue X.509 Certificates to their Immediate Subordinates with appropriate naming constraints as defined in *X.509 PKI* (page 138).
- **Metadata Policy Application**: Apply federation-specific metadata policies with **cascading effect** (Trust Anchor policies override Intermediate policies).
- **Trust Mark Issuance**: Issue Federation Trust Marks attesting Subordinate capabilities and grants, according to the trust framework.

Therefore, Federation Entities MAY be onboarded through different paths:

- **Direct Trust Anchor Onboarding**: Entity directly registers with the Trust Anchor.

- **Intermediate-mediated Onboarding**: Entity registers with an appropriate Intermediate.

8.4.2 Federation Onboarding Prerequisites

Federation Entities MUST comply with the following technical requirements before initiating the onboarding process:

- **Key Generation**: The entities MUST generate at least two key pairs using elliptic curve cryptography as specified in *Cryptographic Algorithms* (page 615):
  - **Federation Key Pair**: Used for signing Entity Configurations and attesting application specific keys. For security best practices and operational continuity, entities SHOULD maintain multiple Federation Entity Keys (at least two) to enable secure key rotation and incident response without impacting entities that have cached Entity Configurations.
  - **Application Specific Key Pair(s)**: Used for entity-specific protocol operations, such as Credential issuance and Credential presentation.
- **Application Specific Key Attestation**: The entities MUST create self-signed X.509 Certificates for their application specific keys using the Federation Entity Private Key.
- **Entity Configuration Preparation**: The entities MUST publish an Entity Configuration (EC) signed with their Federation Entity Private Key at the `/.well-known/openid-federation` endpoint as defined in *The Infrastructure of Trust* (page 109). The EC MUST include:
  - A `jwks` claim containing the Federation Entity Keys in JSON Web Key (JWK) format.
  - An `iss` claim with the Federation Entity Identifier as defined in *Federation Roles* (page 110).
  - A `sub` claim equal to the `iss` claim.
  - `iat` and `exp` claims defining a valid time interval.
  - A `metadata` claim containing entity-specific metadata organized by Metadata Types (see credential-issuer-entity-configuration:Credential Issuer Entity Configuration, *Relying Party Entity Configuration* (page 352), or *Wallet Provider Entity Configuration* (page 305)) with application specific keys included in the metadata `jwks` fields and self-signed X.509 Certificates in the corresponding `x5c` claims.
- **X.509 Certificate Signing Request (CSR)**: The entities MUST prepare a X.509 Certificate Signing Request (CSR) in PKCS #10 format containing **the Federation Entity Key** for X.509 Certificate issuance by the Federation Authority, as defined in *X.509 Certificates Issuance* (page 139).

8.4.3 Key Management Security Requirements

All federation entities SHOULD maintain at least two signing keys attested by the Trust Anchor:

- **Active Key**: Used for current signing operations
- **Backup Key**: Available for immediate activation during incidents or planned key rotation

This dual-key approach enables: - Secure key rotation without service interruption - Rapid incident response when primary keys are compromised - Continuity for entities with cached Entity Configurations - Prevention of validation issues during key transitions

The backup key MUST be: - Registered by the Trust Anchor before deployment - Published in the entity's JWKS alongside the active key - Ready for immediate activation without additional certification steps - Maintained with the same security standards as the active key

8.4.4 Federation Onboarding Procedure

The federation onboarding follows a structured 4-step procedure, it can be performed by the Trust Anchor or an Intermediate.

Note

The following procedure applies to Wallet Providers, Credential Issuers and Relying Parties that wish to perform onboarding in the IT-Wallet federation. The **Federation Authority** is referred to Trust Anchor or Intermediate according to organizational characteristics and federation governance policies.

Note

This section covers the technical registration requirements. All administrative information, such as legal entity validation, regulatory compliance, organizational eligibility, is assumed to have been collected and validated by the Supervisory Body during the administrative registration phase, which is out of scope for this technical specification. Examples of administrative information include: legal entity name, business registration numbers, contact persons, legal compliance documentation, and operational authorizations.

Step 1 - Onboarding Request Submission: Federation Entity initiates the onboarding process by submitting a technical registration request including the following information.

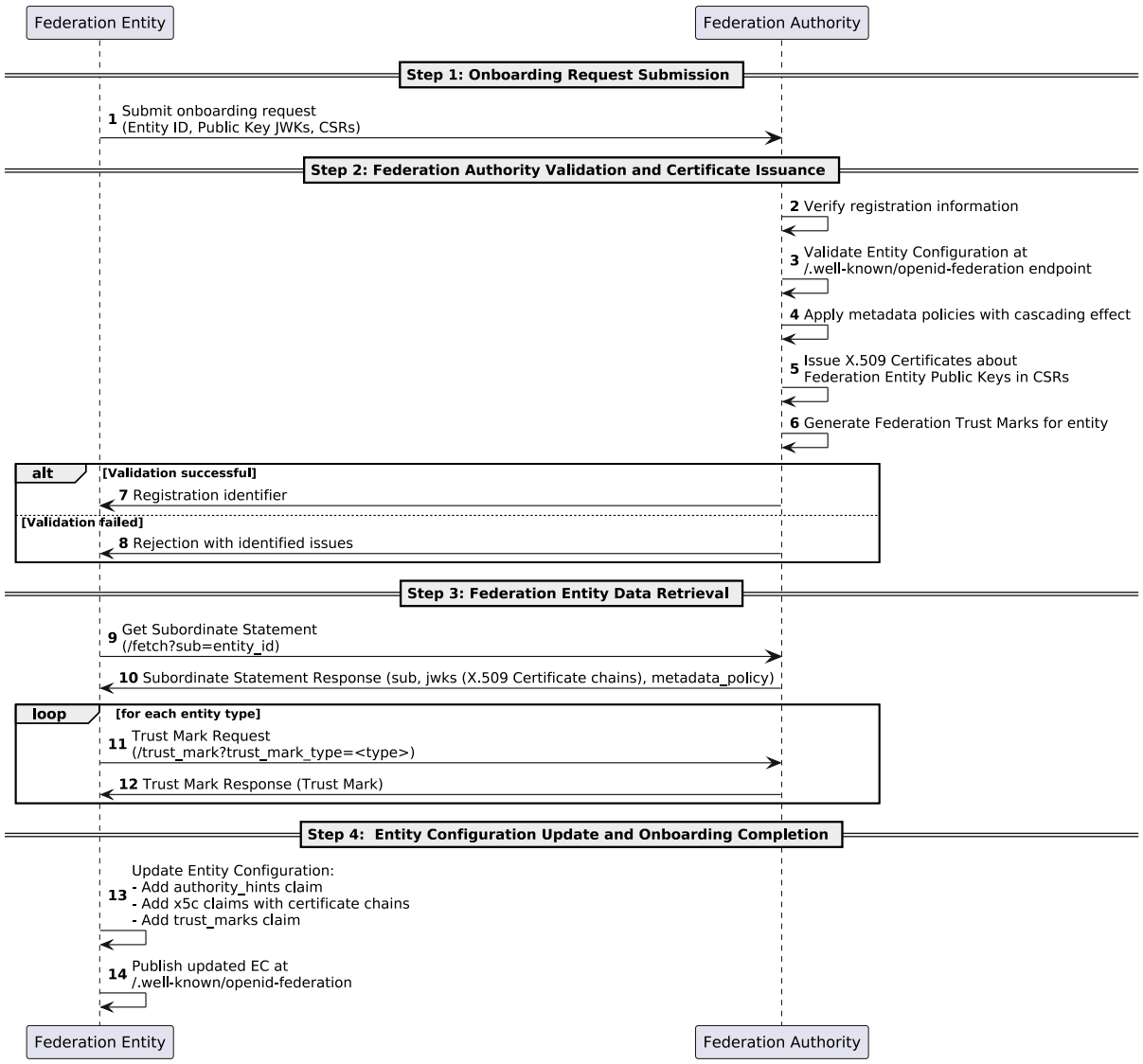


Fig. 8.2: Federation Entity Onboarding Process.

Table 8.5: Federation Onboarding Technical Request Information

Technical Information Category	Requirements and Description
Federation Entity Identifier	**REQUIRED**. A unique URL that identifies the Federation Entity as defined in *Federation Roles* (page 110).
Federation Entity Key (JWK)	**REQUIRED**. Elliptic public key in JSON Web Key format used for signing Entity Configurations and attesting application specific keys, using cryptographic algorithms specified in *Cryptographic Algorithms* (page 615).
X.509 Certificate Signing Request (CSR)	**REQUIRED**. CSR in PKCS #10 format for X.509 Certificate issuance by the Federation Authority. The CSR MUST: • Contain **only the Federation Entity Keys**. • Be signed with the corresponding Federation Entity Private Key. • Define the X.509 Certificate Subject with required attributes as specified in *X.509 Certificates Issuance* (page 139) for Federation Entities.

Warning

Before submitting the technical onboarding request, Federation Entities MUST ensure that their `/.well-known/openid-federation` endpoint publishes a valid Entity Configuration (as defined in *Entity Configuration* (page 117)) signed with their Federation Entity Private Key corresponding to the Federation Entity Key provided in the request. The Entity Configuration MUST already include application specific keys in the metadata with self-signed X.509 Certificates in the `x5c` claims.

A non-normative example of the technical information structure that Federation Entities submit during Step 1 onboarding request:


```
{
    "entity_id": "https://credential-issuer.example.gov",
    "entity_type": "credential_issuer",
    "jwks": {
        "keys": [
            {
                "kid": "k1d2e3f4g5h6i7j8",
                "kty": "EC",
                "crv": "P-256",
                "x": "8w1QwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw",
                "y": "QwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw"
            }
```

```
        ]
    },
    "certificate_signing_requests": {
        "k1d2e3f4g5h6i7j8": "-----BEGIN CERTIFICATE REQUEST-----\nMIIBWTCB/QIBADCBkDELMAkGA1UEB
hMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYD\nVQQHDARSb21hMRYwFAYDVQQKDA1FeGFtcGxlIE9yZzEkMCIGA1UEAwwbY3
JlZGVu\ndGlhbC1pc3N1ZXIuZXhhbXBsZS5nb3YwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC\nAAR8w1QwQwQwQwQwQwQwQ
wQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\nQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw
QwQwQwQwQwQwQw\nQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\nQwQwQwQwQwQwQ
wQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\n-----END CERTIFICATE REQUEST-----"
    },
    "submission_timestamp": "2025-09-25T14:30:00Z"
}
```


The example below shows the decoded content of the CSR:

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: CN=credentials.example.gov, OU=Digital Credentials, O=Example Organization,
        L=Roma, ST=Lazio, C=IT, emailAddress=technical@credentials.example.gov
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                ASN1 OID: prime256v1
                NIST CURVE: P-256
    Signature Algorithm: ecdsa-with-SHA256
```

Note

The CSR Subject attributes MUST comply with the requirements specified in *X.509 Certificates Issuance* (page 139) for Federation Entities.

Note

The Federation Entity Keys contained in the `jwks` object and the public keys contained in the `certificate_signing_requests` object MUST match. The keys are provided in two formats: JWK format and PKCS #10 CSR format for X.509 Certificate issuance. Application specific keys are included only in the Entity Configuration metadata, and they MUST NOT be included in the onboarding request.

Note

The Entity Configuration Endpoint is constructed automatically by appending `/.well-known/openid-federation` to the Federation Entity Identifier (`entity_id`). Federation Entities do not need to specify this endpoint separately in the registration request.

Step 2 - Federation Authority Validation and X.509 Certificate Issuance: Following the onboarding request submission, the **Federation Authority** MUST perform:

- Verification of information provided in the registration request.
- Validation of the Entity Configuration, and the metadata contained in it, published at the entity's `/.well-known/openid-federation` endpoint (as defined in *The Infrastructure of Trust* (page 109)).
- **Metadata Policy Application**: Application of federation-specific metadata policies to the entity's metadata based on organizational characteristics and authorization scope as defined in *Subordinate Statements* (page 123). When onboarded through an Intermediate, both Intermediate and Trust Anchor policies apply, with Trust Anchor policies taking precedence in case of conflicts.
- **X.509 Certificate Issuance**: Certification of the Federation Entity Key with X.509 Certificate issuance using the trust infrastructure detailed in *X.509 Certificates Issuance* (page 139). Intermediates MUST issue X.509 Certificates using appropriate **naming constraints** limiting the use of DNS names and URIs to their subordinates only.

Upon successful validation, the entity receives a Registration Identifier.

Note

If issuance process fails, the requestor entity receives a response with identified issues to be resolved before submitting a new onboarding request.

Note

Each entity can collect its X.509 Certificate Chain by obtaining the Trust Anchor's self-signed X.509 Certificate, which is included within the Entity Configuration's `jwks` object, and by aggregating all the `x5c` values published by its superior entities, from the Trust Anchor down to its immediate superior. This enables the entity to reconstruct the full certificate chain required for validation and trust establishment within the federation.

Step 3 - Federation Entity Data Retrieval: as confirmation of success of the onboarding process, the entity MUST fetch the Subordinate Statement about itself from the immediate Federation Authority using the `/fetch` endpoint as defined in *Federation API endpoints* (page 114).

In this way, the entity receives a X.509 Certificate Chain(s) within `jwks` where:

- The first element is the X.509 Certificate containing the Federation Entity Key (issued by the Federation Authority).
- **For Trust Anchor onboarding**: The second element is the Trust Anchor's self-signed X.509 Certificate for validating the first X.509 Certificate.
- **For Intermediate onboarding**: Additional elements include the Intermediate's X.509 Certificate and the Trust Anchor's self-signed Certificate, forming a complete X.509 X.509 Certificate chain.
- All X.509 Certificates are expressed in DER format and encoded in Base64.

Example X.509 X.509 Certificate chain within `jwks` claim:


```
{
   "keys": [
     {
       "crv": "P-256",
       "kty": "EC",
       "x": "qrJrj3Af_B57sbOIRrcBM7br7wOc8ynj7lHFPTeffUk",
       "y": "1H0cWDyGgvU8w-kPKU_xycOCUNT2o0bwslIQtnPU6iM",
       "kid": "5t5YYpBhN-EgIEEI5iUzr6r0MR02LnVQ0OmekmNKcjY",
                   "x5c": ["MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQsw...", "MIIDQzCC
AuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQsw..."]
     }
   ]
 }
```


Example fetch request:

```
GET /fetch?sub=https%3A%2F%2Fcredentials.example.gov HTTP/1.1
Host: trust-anchor.example.gov
```

Note

If the `/fetch` endpoint responds with status code set to `400` or `404`, the entity MUST resolve the issues described in the response message, before calling the fetch endpoint again.

Furthermore, the entity can retrieve Trust Marks through Trust Mark Endpoint as defined in *Federation API endpoints* (page 114).

Step 4: Entity Configuration Update and Onboarding Completion

Following the fetch request, the **Federation Authority** returns the entity's Subordinate Statement, which is a signed JWT containing:

- **Entity Metadata**: The entity's validated metadata with applied federation policies.
- **Trust Information**: Information about the trust relationship between the Federation Authority and the entity, including the issued *x5c* within the *jwk* object about the Subordinate.
- **Trust Marks**: The Trust Marks issued by the Federation Authority for the entity's specific operational scope and capabilities.

  Example Federation JWK about the Subordinate including the issued X.509 Certificate:

```
{
  "kid": "NsXymfIILEPR5Y0t",
  "kty": "EC",
  "x": "gXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l_g",
  "y": "l-6dcrIrFVdrzoY9cRJv9zNuFOR3MsDz6TSDhB0xEo4",
  "crv": "P-256",
  "x5c": [
    "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQsw..."
  ]
}
```

The Federation Entity MUST complete the onboarding process by:

1. **Update and Publish the Entity Configuration**:

   The content of `/.well-known/openid-federation` endpoint must be updated to include the following claims:

   - Add an `authority_hints` claim with a JSON Array containing the **immediate Federation Authority's** Federation Entity Identifier (Trust Anchor for direct onboarding, or Intermediate for mediated onboarding) as defined in *Federation Roles* (page 110).
   - **Integrate Trust Marks**: Include the Trust Marks from the Subordinate Statement in the Entity Configuration using the `trust_marks` claim as specified in *Entity Configuration Leaves and Intermediates* (page 120).

   Example authority_hints addition:

```
{
  "iat": 1718207217,
  "exp": 1749743216,
  "iss": "https://credentials.example.gov",
```

```
  "sub": "https://credentials.example.gov",
  "authority_hints": ["https://trust-anchor.example.gov"],
  //...
}
```

Example Trust Marks integration:

```
"trust_marks": [
  {
    "trust_mark_type": "https://trust-anchor.example.gov/trust_marks/federation-
entity/credential-issuer",
    "trust_mark": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3Ry
dXN0LWFuY2hvci5leGFtcGxlLmdvdiIsInN1YiI6Imh0dHBzOi8vY3JlZGVudGlhbC1pc3N1ZXIuZXhh
bXBsZS5nb3YiLCJ0cnVzdF9tYXJrX3R5cGUiOiJodHRwczovL3RydXN0LWFuY2hvci5leGFtcGxlLmdv
di90cnVzdF9tYXJrcy9mZWRlcmF0aW9uLWVudGl0eS9jcmVkZW50aWFsLWlzc3VlciIsImlhdCI6MTcw
MDAwMDAwMCwiZXhwIjoxNzMwMDAwMDAwfQ.abc123signature"
  }
]
```

Note

Federation Entities can obtain Trust Marks anytime needed by using the Trust Mark endpoint exposed by a Trust Mark Issuer as defined in *Federation API endpoints* (page 114).

Upon successful completion of Step 4, the **entity onboarding is successfully completed**. The entity is now operational within the IT-Wallet federation and ready for operational activities.

Note

Federation Registry Integration: Upon successful onboarding completion, the entity's Federation Entity Identifier becomes discoverable through the Trust Anchor's entity listing mechanisms (as defined in *The Infrastructure of Trust* (page 109)), indicating active federation participation. The entity becomes part of the federation infrastructure detailed in *Registry Infrastructure* (page 149).

8.4.5 IT-Wallet Federation Trust Marks

Federation Entities receive IT-Wallet Federation Trust Marks during successful onboarding completion. **Trust Marks are issued by the Federation Authority** (Trust Anchor for direct onboarding, Intermediate for mediated onboarding) through the Federation Trust Mark Endpoint and serve as verifiable attestations about compliance with IT-Wallet technical profiles and or authorization policies.

Trust Mark Types and Schema

Entities MAY receive multiple Trust Marks for different purposes. Trust Mark identifiers MUST follow a hierarchical schema that reflects the authorization scope:

`https://<federation_authority_domain>/trust_marks/<purpose>/<entity_type>`

Where:

- `<federation_authority_domain>`: The domain of the issuing Federation Authority.
- `<purpose>`: The Trust Mark purpose. The `federation-entity` purpose is **REQUIRED** for all entities. Additional Trust Mark purposes MAY be supported, such as `authorization_policy` for granular operational scope definitions.
- `<entity_type>`: The recipient entity type (e.g., `credential-issuer`, `relying-party`, `wallet-provider`).

Trust Mark Structure

Trust Marks in Entity Configuration MUST be represented as JSON objects containing the following claims:

Table 8.6: Trust Mark Object Claims (in Entity Configuration)

Claim	Description
trust_mark_type	**REQUIRED**. Identifier for the type of Trust Mark following the schema: `https://<federation_authority_domain>/trust_marks/<purpose>/<entity_type>`.
trust_mark	**REQUIRED**. A signed JSON Web Token representing the Trust Mark issued by the Federation Authority.

The Trust Mark JWT (contained in the `trust_mark` claim above) includes the following claims:

Table 8.7: Trust Mark JWT Claims

Claim	Description
iss	**REQUIRED**. Federation Authority issuing the Trust Mark (immediate superior: Trust Anchor or Intermediate).
sub	**REQUIRED**. Federation Entity Identifier of the subject.
trust_mark_type	**REQUIRED**. Unique Trust Mark identifier, MUST match the `trust_mark_type` claim.
iat	**REQUIRED**. Trust Mark issuance timestamp.
exp	**REQUIRED**. Trust Mark expiration timestamp.
organization_type	**REQUIRED**. Entity organization type (`public` or `private`).

continues on next page

Table 8.7 – continued from previous page

Claim	Description
vat_number	**RECOMMENDED**. VAT number of the entity (typically for private organizations).
legal_identifier	**RECOMMENDED**. Legal registration number or identifier of the entity (e.g., business registration number, tax code).
ipa_code	**RECOMMENDED**. IPA (Indice delle Pubbliche Amministrazioni) code for public sector entities.
organization_name	**RECOMMENDED**. Full name of the Organizational Entity.
email	**RECOMMENDED**. Institutional or PEC email of the organization.
logo_uri	**REQUIRED**. URL pointing to the *Trust Mark* (page 17) for UI/UX purposes.
ref	**OPTIONAL**. URL with additional web information about the Trust Mark.

The following non-normative examples illustrate different Trust Mark JWT contents for federation membership and different authorization policies:


```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://ci.public-authority.gov.example",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/
credential-issuer",
  "iat": 1718207217,
  "exp": 1749743216,
  "organization_type": "public",
  "ipa_code": "pub_001",
  "legal_identifier": "12345678901",
  "organization_name": "Public Authority Services",
  "email": "registry@public-authority.gov.example"
}
```

```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://rental.cars.example.com",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/authorization_poli
cy/relying-party",
  "iat": 1718207217,
  "exp": 1749743216,
  "organization_type": "private",
  "vat_number": "IT12345678901",
  "legal_identifier": "12345678901",
  "organization_name": "Premium Car Rental Services Ltd",
  "email": "compliance@rental.cars.example.com",
  "authorized_claims": ["given_name", "family_name", "driving_privileges"],
  "authorized_credential_types": ["mobile-driving-license"],
  "scope_restrictions": {
    "domains": ["MOBILITY_TRAVEL"],
```

```
    "purposes": ["DRIVING_RIGHTS_VERIFICATION"]
  }
}
```

```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://private-badge.ci.example.com",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/authorization_poli
cy/credential-issuer",
  "iat": 1718207217,
  "exp": 1749743216,
  "organization_type": "private",
  "vat_number": "IT98765432101",
  "legal_identifier": "98765432101",
  "organization_name": "Badge Services Ltd",
  "email": "compliance@rprivate-badge.ci.example.com",
  "authorized_claims": ["given_name", "family_name", "company_id"],
  "authorized_credential_types": ["example-company-badge"],
  "scope_restrictions": {
    "domains": ["EMPLOYMENT"],
    "purposes": ["ACCESS_PERMIT"]
  }
}
```

Federation Entities MUST integrate Trust Marks in their Entity Configuration using the `trust_marks` claim as specified in *Entity Configuration Leaves and Intermediates* (page 120). Entities MAY receive multiple Trust Marks for different authorization scopes.

```
{
  "iss": "https://credentials.example.gov",
  "sub": "https://credentials.example.gov",
  "jwks": {
   // jwks content
  },
  "authority_hints": ["https://trust-anchor.eid-wallet.example.it"],
  "trust_marks": [
    {
      "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-ent
ity/credential-issuer",
      "trust_mark": "eyJhbGciOiJFUzI1NiIsImtpZCI6IlRydXN0QW5jaG9yS2V5SWQiLCJ0eXAiOiJKV1QifQ..."
    }
  ],
  "metadata": {
   // Metadata content
  }
}
```

```
{
  "iss": "https://healthcare-ci.example.gov",
  "sub": "https://healthcare-ci.example.gov",
  "jwks": {
   // jwks content
  },
  "authority_hints": ["https://healthcare.intermediate.eid-wallet.example.it"],
  "trust_marks": [
    {
      "trust_mark_type": "https://healthcare.intermediate.eid-wallet.example.it/trust_marks/federation-entity/credential-issuer",
      "trust_mark": "eyJhbGciOiJFUzI1NiIsImtpZCI6IkhlYWx0aGNhcmVJbnRlcm1lZGlhdGVLZXlJZCIsInR5cCI6IkpXVCJ9..."
    }
  ],
  "metadata": {
   // Metadata content
 }
}
```


Trust Mark Validation

Federation participants validate Trust Mark status through two mechanisms:

1. **Static Validation**: Cryptographic verification using the issuing Federation Authority's public key from the Trust Chain.
2. **Dynamic Validation**: Real-time status verification, against any revocations, using the Federation Authority's `/trust_mark_status` endpoint as defined in *Federation API endpoints* (page 114).

For comprehensive X.509 Certificate management procedures, including chain validation, revocation verification, and lifecycle management, see *X.509 Certificate Management Operations* (page 255).

CHAPTER

NINE

X.509 CERTIFICATE MANAGEMENT OPERATIONS

This section defines the operational procedures for X.509 Certificate management within the IT-Wallet federation, covering X.509 Certificate chain analysis, validation procedures, and revocation verification. These procedures complement the federation onboarding processes and support ongoing X.509 Certificate lifecycle management for all the participants.

For the foundational X.509 PKI infrastructure and certificate issuance procedures, see *X.509 PKI* (page 138).

9.1 Federation PKI Architecture

The IT-Wallet federation operates a hierarchical Public Key Infrastructure where:

- **Trust Anchor**: Acts as Root X.509 Certificate Authority where CA X.509 Certificates MUST NOT exceed **5-year validity period**.
- **Federation Entity X.509 Certificate**: Each federation participant receives an X.509 certificate that operates as a limited sub-CA where X.509 Certificates MUST NOT exceed **2-year validity period**.
- **Protocol X.509 Certificates**: Self-issued X.509 Certificates for internal services where X.509 Certificates SHOULD NOT exceed **1-year validity period**.

Each federation entity MUST expose its Federation Entity X.509 Certificate on a publicly accessible endpoint. The Federation Entity private key serves dual purposes:

1. Self-issuing Protocol X.509 Certificates for internal cryptographic operations (limited sub-CA capability).
2. Acting as the Federation Entity Key for signing Entity Statements.

Note

Federation Leaves can ONLY issue X.509 Certificates about themselves and for application specific purposes. Federation Leaves MUST NOT issue X.509 Certificates for other federation entities, as limited by their Federation Authorities using X.509 Name Constraints extension.

For protocol specific X.509 Certificates, with validity periods exceeding 24 hours, the issuing entity MUST publish and regularly update an X.509 Certificate Revocation List (CRL) on a publicly accessible endpoint.

9.2 X.509 Certificate Chain Structure and Analysis

Federation entities receive X.509 Certificate chains during the onboarding process. Federation entities MUST validate these chains about themselves.

9.2.1 X.509 Certificate Chain Visualization

The following script enables federation entities to:

- Extract X.509 certificate details for verification.
- Analyze X.509 certificate extensions and constraints.
- Verify X.509 certificate hierarchy and relationships

```
#!/bin/bash
# Certificate chain analysis for federation entities
# Array containing certificates in DER format encoded in Base64
certificate_chain=(
    "MIIDyzCCA3GgAwIBAgI..." # Federation Entity Certificate
    "MIIDQzCCAuigAwIBAgI..." # Trust Anchor Certificate
)

# Display first certificate (Federation Entity)
echo "====================================="
echo " Federation Entity Certificate Analysis"
echo "====================================="
echo
echo "${certificate_chain[0]}" | base64 -d > federation_entity.der
openssl x509 -in federation_entity.der -inform DER -text -noout

# Display second certificate (Trust Anchor)
echo "====================================="
echo " Trust Anchor Certificate Analysis"
echo "====================================="
echo
```

```
echo "${certificate_chain[1]}" | base64 -d > trust_anchor.der
openssl x509 -in trust_anchor.der -inform DER -text -noout

# Cleanup temporary files
rm federation_entity.der trust_anchor.der
```

9.2.2 X.509 Certificate Chain Validation

Federation entities MUST validate X.509 certificate chains to ensure proper trust establishment and verify compliance with federation PKI requirements.

A non-normative example of X.509 Certificate chain validation procedure is given below:

```
#!/bin/bash
# Certificate chain validation for federation entities

# Convert DER certificates to PEM format for validation
openssl x509 -inform der -in federation_entity.der -out federation_entity.pem
openssl x509 -inform der -in trust_anchor.der -out trust_anchor.pem

# Validate Trust Anchor certificate (self-signed)
echo "Validating Trust Anchor certificate..."
openssl verify -CAfile trust_anchor.pem trust_anchor.pem

# Validate Federation Entity certificate against Trust Anchor
echo "Validating Federation Entity certificate..."
openssl verify -CAfile trust_anchor.pem federation_entity.pem

# Cleanup
rm federation_entity.pem trust_anchor.pem
```

Federation entities SHOULD verify:

1. **X.509 Certificate Signatures**: Each X.509 Certificate MUST be properly signed by its issuer.
2. **X.509 Certificate Chain Integrity**: Issuer-Subject relationships MUST be valid throughout the X.509 Certificate chain.
3. **X.509 Certificate Validity Periods**: All X.509 Certificates MUST be within their validity periods and MUST comply with federation limits.
4. **X.509 Certificate Extensions**: Basic Constraints and Key Usage MUST comply with federation requirements:
   - Federation Entity X.509 Certificates: `CA:TRUE, pathlen:0` (can only issue certificates about itself).
   - Protocol X.509 Certificates: `CA:FALSE` (cannot issue certificates).

9.3 X.509 Certificate Revocation Management

Federation entities MUST implement X.509 Certificate revocation verification to ensure ongoing trust validation and compliance with federation security requirements.

9.3.1 CRL Distribution and Access

Federation authorities publish X.509 Certificate Revocation Lists (CRL) at publicly accessible endpoints. Federation entities MUST be able to access and process these CRL distributions for revocation verification.

The following procedure enables federation entities to:

- Locate CRL distribution endpoints from X.509 Certificates.
- Download current revocation lists.
- Analyze CRL content and validity periods.

```
#!/bin/bash
# CRL extraction and analysis for federation entities

# Extract CRL URL from certificate CRL Distribution Points extension
crl_url=$(openssl x509 -in certificate.der -inform DER -text -noout | \
          grep "URI:" | sed 's/.*URI://')

echo "CRL Distribution Point: $crl_url"

# Download CRL from distribution point
curl -s -O "$crl_url"
crl_file=$(basename "$crl_url")

# Display CRL information
echo "CRL Content Analysis:"
openssl crl -in "$crl_file" -inform DER -text -noout
```

9.3.2 X.509 Certificate Revocation Verification

Federation entities MUST verify X.509 certificate revocation status by checking X.509 certificate serial numbers against current X.509 Certificate Revocation Lists.

Federation entities SHOULD implement automated revocation checking for:

- **Federation Entity X.509 Certificates**: Verify own X.509 certificate status periodically.
- **Peer Entity X.509 Certificates**: Validate X.509 Certificates of other federation participants.

- **Trust Chain Validation**: Ensure entire X.509 certificate chains remain valid.

Below a bash script for X.509 certificate revocation status verification is given as a non-normative example:

```
#!/bin/bash
# Certificate revocation verification for federation entities

# Extract certificate serial number
certificate_serial=$(openssl x509 -in certificate.der -inform DER -noout -serial | \
                  cut -d= -f2)

# Normalize serial number (remove leading zeros, convert to lowercase)
normalized_serial=$(echo "$certificate_serial" | sed 's/^0*//' | tr '[:upper:]' '[:lower:]')

echo "Certificate Serial Number: $normalized_serial"

# Extract CRL URL and download
crl_url=$(openssl x509 -in certificate.der -inform DER -text -noout | \
          grep "URI:" | sed 's/.*URI://')
curl -s -O "$crl_url"
crl_file=$(basename "$crl_url")

# Validate CRL signature against Trust Anchor
echo "Validating CRL signature..."
openssl crl -in "$crl_file" -inform DER -noout -text -CAfile trust_anchor.pem

# Extract revoked serial numbers from CRL
revoked_serials=$(openssl crl -in "$crl_file" -inform DER -text -noout | \
                  grep 'Serial Number' | \
                  sed 's/.*Serial Number: //' | \
                  sed 's/^0*//' | \
                  tr '[:upper:]' '[:lower:]')

# Check if certificate is revoked
if echo "$revoked_serials" | grep -q "$normalized_serial"; then
    echo "Certificate Status: REVOKED"
    exit 1
else
    echo "Certificate Status: VALID"
    exit 0
fi
```

9.4 X.509 Certificate Management Best Practices

9.4.1 X.509 Certificate Validation Integration

Federation entities SHOULD integrate X.509 Certificate validation procedures into their standard federation operations:

1. **Entity Configuration Updates**: Verify X.509 Certificate chains when processing authority hints and X.509 Certificate updates.
2. **Trust Chain Construction**: Validate all X.509 Certificates during Trust Chain building procedures.
3. **X.509 PKI Operations**: Perform X.509 Certificate revocation checks using CRL endpoints.
4. **Protocol Certificate Management**: Validate self-issued protocol specific X.509 Certificate for internal services.
5. **Periodic Validation**: Implement regular X.509 Certificate and CRL validation schedules.

9.4.2 Diagnostic and Troubleshooting

Federation entities MUST implement diagnostic procedures to identify and resolve X.509 Certificate-related issues:

- **X.509 Certificate Validation**, including:
  - **Authority Key Identifier Mismatches**: CRL Authority Key Identifier does not match Trust Anchor Subject Key Identifier.
  - **Trust Anchor X.509 Certificate Rotation**: Outdated Trust Anchor X.509 Certificate causing validation failures.
  - **Serial Number Format Issues**: Serial number normalization problems in revocation checking.
- **CRL Validation Failure**: When CRL validation fails, federation entities SHOULD:
  1. **Verify Trust Anchor X.509 Certificate**: Ensure current Trust Anchor X.509 certificate is being used.
  2. **Check Authority Key Identifier**: Compare CRL Authority Key Identifier with Trust Anchor Subject Key Identifier.
  3. **Validate CRL Signature**: Verify CRL is properly signed by expected issuing authority.

4. **Update Trust Anchor X.509 Certificate**: Download updated Trust Anchor X.509 certificate if rotation has occurred.

- **Endpoint Accessibility Verification**: Federation entities SHOULD implement connectivity testing for certificate infrastructure endpoints.

The following non-normative example provides a script for Federation X.509 certificate infrastructure connectivity test:

```
#!/bin/bash
# Federation certificate infrastructure connectivity test

# Test Trust Anchor certificate endpoint
ta_cert_url="https://trust-anchor.eid-wallet.example.it/pki/ta.cer"
if curl -f -s "$ta_cert_url" > /dev/null; then
    echo "Trust Anchor certificate endpoint: ACCESSIBLE"
else
    echo "Trust Anchor certificate endpoint: FAILED"
fi

# Test CRL distribution endpoints
ta_crl_url="https://trust-anchor.eid-wallet.example.it/pki/ta.crl"
if curl -f -s "$ta_crl_url" > /dev/null; then
    echo "Trust Anchor CRL endpoint: ACCESSIBLE"
else
    echo "Trust Anchor CRL endpoint: FAILED"
fi
```

9.4.3 X.509 Certificate Lifecycle Coordination

Federation entities MUST coordinate X.509 Certificate management with federation lifecycle procedures following the established validity periods:

- **X.509 Certificate Renewal**: Align X.509 certificate renewals with Entity Configuration updates and Trust Mark refresh cycles, according to the federation limits defined in *Federation PKI Architecture* (page 255).
- **Key Rotation**: Coordinate Federation Entity Key rotation with X.509 certificate renewal procedures.
- **CRL Management**: For Protocol X.509 Certificates with validity > 24 hours, maintain current CRL publication.
- **Federation Exit**: Ensure proper X.509 Certificate revocation during voluntary or supervisory body-initiated federation exit.

9.5 Key Rotation Procedures

Federation entities MUST implement key rotation mechanisms to maintain security posture and respond to security events. This section defines the operational procedures for detecting and responding to superior key rotations and for subordinate key renewal.

9.5.1 Superior Key Rotation

Subordinate entities MUST perform periodic verification of their Trust Chain to detect superior key rotations and ensure continued federation membership.

Verification Requirements

Federation Subordinates MUST:

1. Periodically fetch their own Subordinate Statement from immediate superior using the fetch endpoint defined in *Federation API endpoints* (page 114). This SHOULD be made within 24 hours.
2. After fetching, verify the validity of the Superior's X.509 certificate in the `x5c` field, including checking for revocation or suspension status.
3. Update local Entity Configuration when superior certificates change.

Detection Mechanisms

Subordinates MAY use one or more of the following mechanisms to detect changes:

- **CRL/OCSP Verification**: Monitor Certificate Revocation Lists or Online Certificate Status Protocol responses to detect revocation of superior certificates as defined in *X.509 Certificate Revocation* (page 145). Certificate revocation indicates superior key rotation is in progress.
- **Fetch Endpoint**: Retrieve Subordinate Statement via `GET {superior}/fetch?sub={subordinate_entity_id}` to obtain current certificates.
- **Events Endpoint**: Monitor `GET {superior}/federation_subordinate_events_endpoint?sub={subordinate_entity_id}` for `jwks_update` events. The Federation Subordinate Events endpoint returns historical track of registration events as defined in *Federation API endpoints* (page 114).

Subordinates SHOULD implement automated scheduling for:

- Daily fetch operations to retrieve current Subordinate Statement.
- X.509 Certificate chain validation after each fetch.
- Automatic Entity Configuration updates when certificate changes detected.

- Event monitoring for proactive change detection.

When a superior entity (Trust Anchor or Intermediate) rotates its Federation Entity Key, all subordinate X.509 certificates MUST be reissued.

Rotation Procedure

1. **Key Revocation**: Superior publishes Certificate Revocation List (CRL) or Historical Keys (HK) entry revoking the old key as defined in *X.509 Certificate Revocation* (page 145).
2. **Entity Configuration Update**: Superior publishes new Entity Configuration signed with new Federation Entity Key.
3. **X.509 Certificate Reissuance**: Superior reissues X.509 certificates to all subordinates using new key.
4. **Certificate Publication**: Superior makes new certificates available on fetch endpoint.
5. **Subordinate Detection**: Subordinates detect change during periodic fetch (within 24 hours).
6. **Certificate Update**: Subordinates update `x5c` field in their Entity Configuration with new certificate.

Certificate Priority

When multiple X.509 certificates exist for the same subordinate public key, the certificate in the Subordinate Statement MUST take priority over the certificate in the subordinate's Entity Configuration.

Note

A superior entity cannot independently initiate a Subordinate Key rotation process. The Subordinate Key Rotation mechanism is always initiated by the Subordinate entity, see Section *Subordinate Key Rotation* (page 263).

9.5.2 Subordinate Key Rotation

Subordinates rotate their Federation Entity Keys by submitting Certificate Signing Requests to their immediate superior following the same procedure as initial onboarding.

Rotation Procedure

1. **New Key Generation**: Subordinate generates new Federation Entity Key pair.
2. **CSR Generation**: Subordinate generates PKCS #10 Certificate Signing Request for new key as defined in *Entity Onboarding* (page 225).

3. **CSR Submission**: Subordinate submits CSR to superior entity using the same onboarding endpoint.
4. **Certificate Issuance**: Superior validates CSR and issues new X.509 certificate.
5. **Certificate Publication**: Superior makes new certificate available on fetch endpoint.
6. **Certificate Retrieval**: Subordinate fetches updated Subordinate Statement containing new certificate.
7. **Entity Configuration Update**: Subordinate adds new JWK with new certificate to Entity Configuration `jwks` array.

Note

During key rotation, subordinates SHOULD maintain both old and new keys temporarily to ensure service continuity. The old key SHOULD be removed only after confirming successful propagation of the new key throughout the federation.

Entity Lifecycle Management

This section provides technical implementation procedures for entity lifecycle management. For high-level lifecycle concepts and business processes, see *Entity Lifecycle Management* (page 27).

While administrative data update MUST follow governance processes that are out of scope of this technical specification, technical configuration updates are described in the following sections.

9.6 Technical Configuration Updates

Technical updates affecting federation protocol operations MUST follow specific procedures for:

- **X.509 Certificate Renewal**
  1. **Pre-renewal Preparation**: The Entity MUST generate a new CSR with updated X.509 certificate information.
  2. **Renewal Request**: The Entity MUST submit renewal request with new CSR following the same technical procedure as initial onboarding.
  3. **X.509 Certificate Integration**: The Entity MUST update its Entity Configuration with new X.509 certificate chain in `x5c` parameter.
  4. **Trust Chain Validation**: The Entity MUST verify updated Trust Chain through `/resolve` endpoint.

5. **Registry Update**: The Entity SHOULD confirm updated entity information in Trust Anchor `/list` endpoint.

- **Key Rotation**
  1. **Key Activation**: In case of incident or planned rotation, the Entity MUST activate an alternative Federation Entity Key for signing operations.
  2. **New Key Generation**: The Entity MUST generate a new Federation Entity Public Key pair to serve as an additional key.
  3. **Parallel Key Publication**: The Entity MUST publish all available keys in Entity Configuration `jwks` claim during transition period.
  4. **X.509 Certificate Request**: The Entity MUST request new X.509 certificate for new key following standard procedure.
  5. **Gradual Migration**: The Entity MUST update Entity Configuration to use the activated key for signing while maintaining all keys for verification.
  6. **Old Key Deprecation**: The Entity MUST remove the old key from Entity Configuration after validation period, maintaining the alternative key and new key.
- **Metadata Updates**
  - **Endpoint Changes**: The Entity MAY update service endpoints in entity-specific metadata.
  - **Capability Updates**: The Entity MAY modify supported protocols, algorithms, or service capabilities within the constraints defined by this IT-Wallet implementation profile.

All technical updates MUST be validated through:

1. **Entity Configuration Validation**: The Entity MUST verify updated EC structure and content.
2. **Trust Chain Resolution**: The Entity MUST confirm `/resolve` endpoint returns valid Trust Chain.
3. **Federation Status**: The Entity MUST verify entity operational status in federation registry.
4. **Integration Testing**: The Entity SHOULD test federation protocol operations with updated configuration.

9.7 Technical Federation Exit Procedures

For business context on federation exit processes, see *Federation Exit and Removal Processes* (page 28). This section covers technical implementation procedures.

9.7.1 Voluntary Exit - Technical Deactivation

1. **X.509 Certificate Revocation Request**: The Entity MUST submit a X.509 Certificate revocation request to Federation Authority with revocation reason. The request MUST be signed with the Federation Entity Private Key corresponding to the X.509 Certificate being revoked to prove the legitimacy of the revocation request.
2. **CRL Update Verification**: The Federation Authority MUST revoke the Entity's X.509 Certificate and the Entity MUST verify they appear in updated X.509 Certificate Revocation List.
3. **Subordinate Statement Removal**: The Federation Authority MUST completely remove the Entity's Subordinate Statement from the Federation Registry to prevent any trust relationship validation, and therefore remove any reference to that Entity in the listing endpoint, resolve endpoint or trust marked listing endpoint.
4. **Entity Configuration Deactivation**: The Entity MUST deactivate its Entity Configuration. The Entity MAY either:
    a. Remove the Entity Configuration completely from the `/.well-known/openid-federation` endpoint (returning HTTP 404), OR
    b. Keep the Entity Configuration as expired (with `exp` claim in the past). It therefore MUST NOT update it with fresh timestamps.
5. **Registry Status Update**: The Entity SHOULD verify removal from Federation Registry, also verifying the Trust Mark status using the Trust Mark Status endpoint.

Non-normative example of X.509 Certificate revocation request following **RFC 3280** format:

```
X.509 Certificate Revocation Request:
Subject: CN=credentials.example.gov, OU=Digital Credentials, O=Example Organization, L=Roma,
ST=Lazio, C=IT, emailAddress=technical@credentials.example.gov
X.509 Certificate Serial Number: 987654321
Revocation Reason: cessation_of_operation (5)
Revocation Date: 2025-12-31T23:59:59Z

Request signed with Federation Entity Private Key corresponding to:
Public Key Algorithm: id-ecPublicKey
ASN1 OID: prime256v1
NIST CURVE: P-256
Key ID: NsXymfIILEPR5Y0t
```

```
Note: The CRR MUST be signed with the same private key that corresponds to the
X.509 certificate being revoked to authenticate the revocation request.
```

Example CRR in DER format (Base64 encoded):

```
-----BEGIN CERTIFICATE REVOCATION REQUEST-----
MIIBVjCB/wIBADBpMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNV
BAcMBFJvbWExGjAYBgNVBAoMEUV4YW1wbGUgT3JnYW5pemF0aW9uMR8wHQYDVQQD
DBZjcmVkZW50aWFscy5leGFtcGxlLmdvdjBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABIBgZ4HBgUCNXwY5LJSlKzm7gXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq
0ob4l_gslT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrYwDQYJKoZIhvcNAQ
kEAwIJrRLl1VR987654321gBgJKwYBBQUHAgEWHGh0dHBzOi8vZXhhbXBsZS5vcmcv
cG9saWN5MAoGCCqGSM49BAMCA0gAMEUCIQC9h3Y6hFgd7zUzZyBrQ3jJ8HmVF2Qa
-----END CERTIFICATE REVOCATION REQUEST-----
```

9.7.2 Supervisory Body Removal - Technical Implementation

1. **Emergency X.509 Certificate Revocation**: The Federation Authority MUST immediately revoke X.509 Certificate with appropriate reason code (e.g., "Key Compromise", "Cessation of Operation").
2. **CRL Emergency Update**: The Trust Anchor MUST publish updated CRL within emergency timeframe.
3. **Subordinate Statement Removal**: The Federation Authority MUST immediately stop issuing the Entity's Subordinate Statements using the fetch endpoint, and or any reference to that Entity using the listing endpoint, or the trust marked listing endpoint, or the resolve endpoint.
4. **Entity Configuration Invalidation**: The Entity's Configuration at `/.well-known/openid-federation` becomes invalid due to X.509 Certificate revocation (signature verification fails).
5. **Trust Chain Invalidation**: Trust Chain resolution MUST return error status for affected entity.
6. **Service Endpoint Isolation**: Federation infrastructure MUST block access to federation service endpoints.

Example emergency revocation verification:

```
# Check emergency CRL update
curl -o emergency.crl https://trust-anchor.eid-wallet.example.it/pki/ta-sub.crl
openssl crl -in emergency.crl -text -noout | grep "Last Update"

# Verify Trust Chain resolution fails
```

```
curl "https://trust-anchor.eid-wallet.example.it/resolve?sub=https%3A//suspended-entity.example
.it&trust_anchor=https%3A//trust-registry.eid-wallet.example.it"
# Expected: HTTP 404 or specific error response
```

Component-Level Technical Modifications

Specific technical components MAY be modified while maintaining federation membership:

```
{
  "iss": "https://ci.example.it",
  "sub": "https://ci.example.it",
  "jwks": {
    // jwks content
  },
  "metadata": {
    "openid_credential_issuer": {
      "credential_endpoint": "https://ci.example.it/credential",
      "credentials_supported": [ ]
      // removed: "batch_credential_endpoint" for maintenance
    }
  }
}
```

The Entity MUST follow these steps for component modifications:

1. **Entity Configuration Update**: The Entity MUST modify metadata to reflect component changes.
2. **Trust Chain Revalidation**: The Entity MUST verify updated configuration through `/resolve` endpoint.
3. **Service Testing**: The Entity SHOULD test remaining federation protocol operations.
4. **Registry Verification**: The Entity SHOULD confirm updated capabilities in federation registry.

Post-Exit Technical Obligations

Entities that exit the federation MUST maintain the following for regulatory compliance:

1. **Historical Entity Configuration**: The Entity MUST maintain `/.well-known/openid-federation` endpoint accessibility for audit purposes.
2. **X.509 Certificate Chain Archive**: The Entity MUST keep X.509 Certificate chains accessible for existing Credential verification (minimum 7 years).
3. **Audit Log Preservation**: The Entity MUST archive federation protocol logs per regulatory requirements.

CHAPTER

TEN

ENTITIES

10.1 Wallet Solution

The Wallet Solution is issued by the Wallet Provider in the form of a mobile app and services, such as web interfaces. The mobile app serves as the primary interface for Users, allowing them to hold their Digital Credentials and interact with other participants of the ecosystem, such as Credential Issuers and Relying Parties. These Digital Credentials are a set of data that can uniquely identify a natural or legal person, along with other Qualified and non-qualified Electronic Attestations of Attributes, also known as QEAAs and EAAs respectively, or (Q)EAAs for short. Once a User installs the mobile app on their device, such an installation is referred to as a Wallet Instance for the User. By supporting the mobile app, the Wallet Provider ensures the security and reliability of the entire Wallet Solution, as it is responsible for issuing the Wallet Instance Attestation (WIA) and Key Attestation (KA). WIA proves the authenticity and integrity of the Wallet Instance, while KA provides evidence that the keys used for Credential key binding are securely stored in a trustworthy WSCD. Additionally, KA confirms that the Wallet Unit has not been revoked.

The following diagram depicts the Wallet Solution High Level Architecture.

10.1.1 Wallet Solution Requirements

This section lists the requirements about Wallet Providers and Wallet Solutions with their Wallet Instances, as well as the corresponding Wallet Instance Attestation, Key Attestation and the secure storage component (WSCD or Keystore).

- The Wallet Solution MUST adhere to the specifications set by this document for obtaining Personal Identification (PID) and (Q)EAAs.
- The Wallet Provider MUST expose a set of endpoints, exclusively available to its Wallet Solution instances, supporting the core functionalities of the Wallet Instances.
- The Wallet Instance MUST periodically reestablish trust with its Wallet Provider, obtaining a fresh Wallet Instance Attestation (*WP_018* (page 739)).

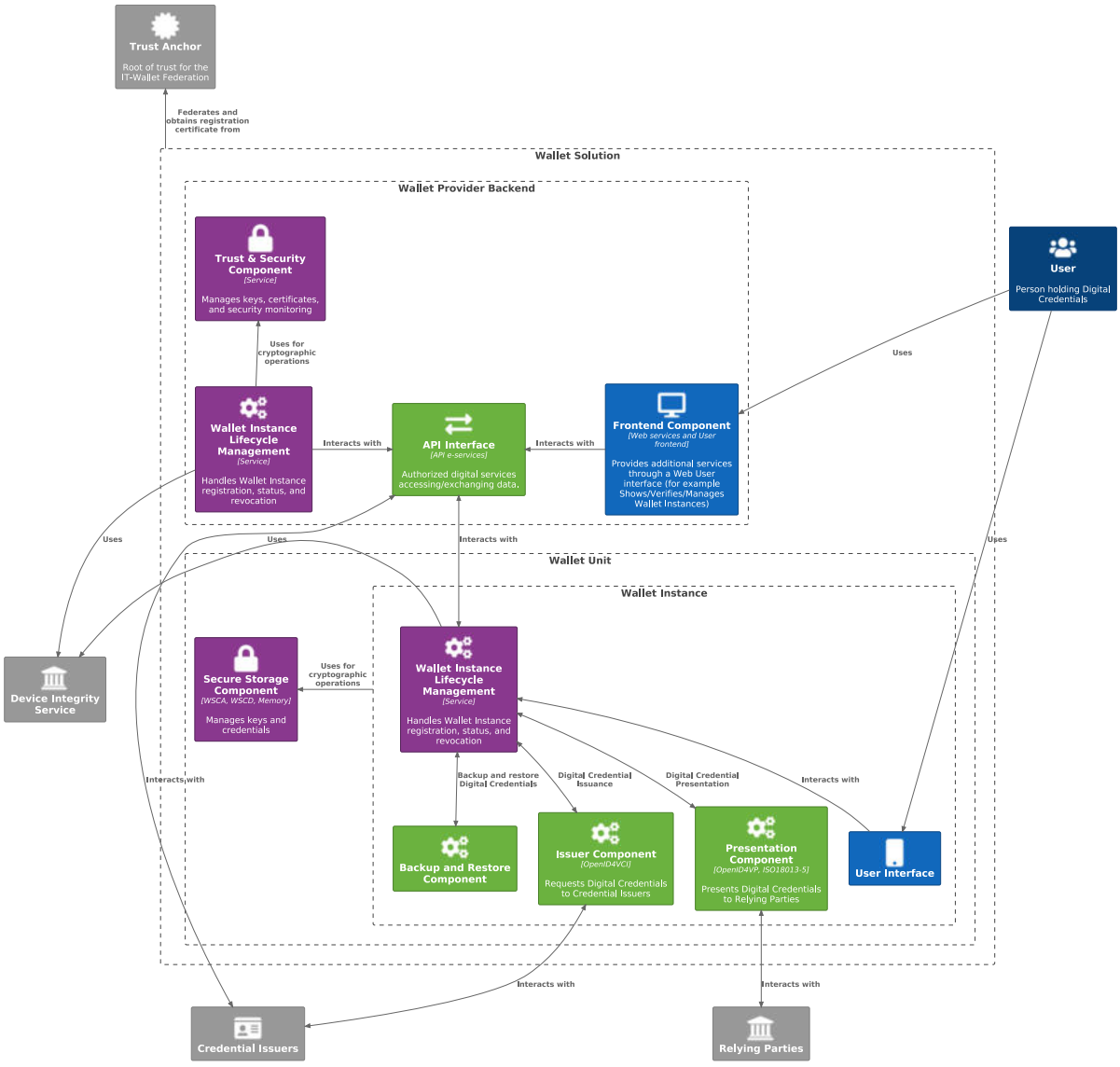


Fig. 10.1: Wallet Solution High Level Architecture.

- The Wallet Instance MUST establish trust with other participants of the Wallet ecosystem, such as Credential Issuers. In case of Credential Issuers, Wallet Instance presents both Wallet Instance and Key Attestations.
- The Wallet Instance MUST be compatible and functional on both Android and iOS operating systems and available on the Play Store and App Store, respectively (*WP_015* (page 739)).
- The Wallet Instance MUST provide a mechanism to verify the User's actual possession and full control of their personal device.
- The Wallet Instance MUST provide Users with an up-to-date list of Relying Parties with which the User has established a connection and, where applicable, all data exchanged;
- The Wallet Instance MUST provide Users with a mechanism to request the erasure of personal attributes by a Relying Party pursuant to Article 17 of Regulation (EU) 2016/679, and to log each Erasure Request made.

Note

There is no strict one-to-one mapping between the requirements in this section and the test cases in *Wallet Provider Test Matrix* (page 736). Some requirements are expressed at too high a level to be represented as atomic test cases, while others are already addressed in greater detail within related flows (e.g., *Wallet Instance Attestation Issuance* (page 285)).

Wallet Instance Attestation Requirements

Wallet Instance Attestation contains information regarding the security level of the device hosting the Wallet Instance. It primarily proves the **authenticity**, **integrity**, **security**, and in general the **trustworthiness** of a particular Wallet Instance.

The requirements for the Wallet Instance Attestation are defined below:

- The Wallet Instance Attestation MUST provide all the relevant information to attest to the **integrity** and **security** of the device where the Wallet Instance is installed (*WP_019* (page 739)).
- The Wallet Instance Attestation MUST be signed by the Wallet Provider that has authority over and is the owner of the Wallet Solution, as specified by the overseeing Registration Authority. This ensures that the Wallet Instance Attestation uniquely links the Wallet Provider to this particular Wallet Instance (*WP_020* (page 739)).
- The Wallet Provider MUST periodically evaluate and guarantee the integrity, the authenticity, and the genuineness of the Wallet Instance. The Wallet Provider verifies the Wallet

Instance using the most secure flow made available by OS Provider's API, such as the *Play Integrity API* for Android and *App Attest* for iOS (*WP_011* (page 737)).

- The Wallet Instance Attestation MUST be securely bound to the Wallet Instance's ephemeral public key (*WP_019b* (page 739)).
- The Wallet Instance Attestation MAY be used multiple times during its validity period, allowing for repeated authentication and authorization without the need to request new attestations with each interaction. However, it is RECOMMENDED that Wallet Instances avoid using the same attestation repeatedly, due to privacy concerns such as linkability between different interactions.
- The Wallet Instance Attestation MUST have a time-to-live of less than 24 hours: the difference between the expiration time (`exp`) and the issuance time (`iat`) MUST be less than 24 hours, after which it MUST no longer be considered valid (*WP_028* (page 739), *WP_144* (page 767)).
- The Wallet Instance Attestation MUST NOT be issued by the Wallet Provider if the authenticity, integrity, and genuineness of the Wallet Instance requesting it cannot be guaranteed (*WP_019a* (page 739)).
- Each Wallet Instance SHOULD be able to request multiple Wallet Instance Attestations using different cryptographic public keys associated with them.
- The Wallet Instance Attestation MUST NOT contain information about the User in control of the Wallet Instance (*WP_029b* (page 739)).
- The Wallet Instance MUST secure a Wallet Instance Attestation as a prerequisite for transitioning to the Operational state, as defined by EIDAS-ARF.
- A Wallet Provider SHALL ensure that a non-revoked Wallet Unit at all times presents a temporally valid and non-revoked Wallet Instance Attestation to a PID Provider or Attestation Provider during the issuance process of a PID or attestation. Note: This requirement applies to both device-bound and non-device-bound attestations, as defined by EIDAS-ARF.
- A Wallet Unit SHALL present a Wallet Instance Attestation only to a PID Provider or Attestation Provider, as part of the issuance process of a PID or an attestation, and not to a Relying Party or any other entity.

Note

Throughout this section, the services used to attest genuineness of the Wallet Instance and the device in which it is installed are referred to as **Device Integrity Service API**. The Device Integrity Service API is considered in an abstract fashion and it is assumed to be

a service provided by a trusted third party (i.e., the OS Provider's API) which is able to perform integrity checks on the Wallet Instance as well as on the device where it is installed.

Key Attestation Requirements

Key Attestation contains information to ensure that keys used for Digital Credential key binding are stored in a **trustworthy** WSCD or Keystore. It also provides a method to authenticate that storage component with the Credential Issuer and verifies that the Wallet Unit has not been revoked.

The requirements for the Key Attestation are defined below:

- The Key Attestation SHALL provide a PID Provider or Attestation Provider with information about the capabilities of the WSCA and WSCD of the Wallet Unit, such that they are able to take a well-grounded decision on whether to issue a PID or attestation to the Wallet Unit.
- The Key Attestation SHALL enable PID Providers and Attestation Providers to verify the authenticity and revocation status of the Wallet Unit.
- A Wallet Provider SHALL ensure that a non-revoked Wallet Unit at all times can present a Key Attestation, when requested by a PID Provider or Attestation Provider.
- During issuance of a PID at Level of Assurance High, the Wallet Unit SHALL provide the PID Provider with a valid Key Attestation (KA) describing the WSCD that generated the new PID private key. PID keys at LoA High MUST be generated and stored only in a WSCD. The current implementation profile does not include PID High issuance; IT-Wallet ID and (Q)EAA keys MAY be generated and stored in a Keystore.
- During issuance of device-bound attestation, a Wallet Unit SHALL retrieve the requirements of the Attestation Provider regarding key storage by the WSCA/WSCD or Keystore from the Issuer metadata (as specified in OpenID4VCI). The Wallet Unit SHALL determine which of its WSCA/WSCD or Keystore instances, if any, comply with these requirements. If a compliant WSCA/WSCD or Keystore is available to the Wallet Unit, the Wallet Unit SHALL provide the Attestation Provider with a valid KA describing the selected WSCA/WSCD or Keystore. Note: A KA describes the properties of the WSCA/WSCD or Keystore and contains one or more public key(s) corresponding to private key(s) generated by and stored in that WSCA/WSCD or Keystore.
- If a Wallet Unit contains multiple WSCAs, it SHALL, internally and securely, keep track of which PIDs and attestations are bound to which WSCA.
- A Wallet Unit SHALL present a Key Attestation only as part of the issuance of a PID or a key-bound attestation.

- The Key Attestation SHALL enable PID Providers to request a Wallet Provider to revoke a Wallet Unit, by including an identifier for the Wallet Unit in the KA (e.g., a URI and index to an Attestation Status Lis). The Wallet Provider SHALL ensure that this Wallet Unit identifier does not enable tracking of the User.
- The Key Attestation MUST contain one or multiple attested Credential's public key that are coming from the same WSCD or Keystore.
- The Key Attestation MUST be signed by the Wallet Provider that has authority over and is the owner of the Wallet Solution, as specified by the overseeing Registration Authority. Wallet Providers SHALL ensure that the certificates they use for signing KAs and WIAs comply with all applicable requirements in ETSI TS 119 412-6, in particular Clause 5.
- An Attestation Provider issuing non-device-bound attestations SHALL indicate in its Credential Issuer metadata that it does not need a KA. A Wallet Unit SHALL NOT send a KA to an Attestation Provider when requesting a non-device-bound attestation. Note: A Wallet Unit sends a WIA to the Attestation Provider regardless of whether the attestations it issues are device-bound or not.
- A Wallet Provider SHALL ensure that the presentation of a KA is cryptographically bound to the specific context it is intended to be used in. Note: As specified in OpenID4VCI, this is achieved by letting the signed KA itself contain a nonce provided by the PID Provider or Attestation Provider during the issuance process. Alternatively, the Wallet Unit presents the KA along with a Proof-of-Possession consisting of a signature over that nonce, created by the private key corresponding to one of the public keys attested in the KA.
- During issuance of a PID or a device-bound attestation, the PID Provider or Attestation Provider SHALL verify the KA in accordance with the requirements in OpenID4VCI Appendix F.4.
- During issuance of a PID or a device-bound attestation, the PID Provider or Attestation Provider SHALL receive a proof that the Wallet Unit possesses the private keys corresponding to all public keys in the KA.
- The WSCA, WSCD or Keystore MUST NOT allow export of User private keys in clear. If a device reports a private key as exportable, the Wallet Provider MUST reject the Key Attestation and MUST NOT activate the Wallet Instance (*WP_014b* (page 739)).
- A Wallet Provider SHALL consider all relevant factors, including offline usage, interoperability, and the risk of a KA becoming a vector to track the User, when deciding on the validity period of a KA.
- The Key Attestation MUST NOT be issued by the Wallet Provider if the trustworthiness of the WSCD or Keystore is not guaranteed. In this case, the Wallet Instance MUST be revoked.

WSCD and Keystore Requirements

To guarantee the utmost security, the cryptographic keys associated with a Wallet Instance (for example keys used to generate the Wallet Instance Attestation) MUST be securely generated and stored in a **Keystore** or a **WSCD**, as defined in *Defined Terms and Acronyms* (page 639). The User MUST retain exclusive control of those private keys (Sole Control). The Keystore or WSCD MUST require User authentication (Wallet unlock: PIN or biometric) before any signature or other private-key operation (*WP_014c* (page 739)).

The following approaches MAY be used:

- **Local Internal Keystore**: hardware-backed key storage native to the User's device, such as the Secure Enclave on iOS, or the Trusted Execution Environment (TEE) and StrongBox on Android. This is the component named Local Internal WSCD in some ARF texts. It is a Keystore, not a High-certifiable WSCD.
- **Local External WSCD**: hardware external to the User's device, such as a smart card compliant with *GlobalPlatform* and supporting *JavaCard*, certifiable as a WSCD.
- **Remote WSCD**: a remote Hardware Security Module (HSM) certifiable as a WSCD.
- **Local Hybrid WSCD**: a pluggable internal hardware component within the User's device, such as an *eUICC* that adheres to *GlobalPlatform* standards and supports *JavaCard*.
- **Remote Hybrid WSCD**: a local component mixed with a remote service.

Private keys of an IT-Wallet ID and of a (Q)EAA MAY be stored in a Local Internal Keystore (eIDAS Substantial). Private keys of a PID at LoA High MUST be stored only in a WSCD. A Local Internal Keystore MUST NOT be treated as evidence of LoA High. Private keys bound to a WIA MUST be generated and stored in the same Keystore or WSCD that the WIA attests (*WP_014* (page 739)–*WP_014e* (page 739)).

A Mobile Relying Party Instance that does not store User identity keys MAY use a Local Internal Keystore and is not required to use a High-certifiable WSCD (*WP_014f* (page 739)).

Digital Credentials whose bound private keys are stored in a local Keystore or local WSCD MAY be presented in proximity or otherwise offline. Digital Credentials whose bound private keys are stored in a remote WSCD MUST NOT be presented offline (*WP_160* (page 739)).

Warning

At the current stage, the implementation profile defined in this document supports only the **Local Internal Keystore** (*WP_014* (page 739)). Future versions of this specification MAY include other approaches depending on the required Authenticator Assurance Level (*AAL*).

For more detailed information, please refer to *Wallet Instance Initialization and Registration* (page 285), *Wallet Instance Attestation Issuance* (page 285), and *Key Attestation Issuance* (page 288) of this document.

10.1.2 Wallet Solution Components

Note

Tests related to the Wallet Solution components (Wallet Backend, and Wallet Unit) are summarized in *WP_012* (page 737), and *WP_013* (page 739), accordingly.

Wallet Backend

Frontend Component

The Frontend Component MUST provide a web-based User interface for Wallet Instance management, offering functionality to:

- Display and verify Wallet Instances and their status.
- Manage Wallet Instance lifecycle (e.g., revocation).
- Provide User support and documentation.

API Interface

This component MUST:

- forward the request from the Frontend Component or the Wallet Instance to the Wallet Instance Lifecycle Management component.
- use PDND according to rules in Section *e-Service PDND* (page 675) to be notified by the PID/IT-Wallet ID Provider of the need to revoke the Wallet Instance and delete the User's account due to the User's death.

Wallet Instance Lifecycle Management

This component MUST handle:

- Wallet Instance Registration (detailed in *Wallet Instance Initialization and Registration* (page 285)).
- Wallet Instance Attestation Issuance (detailed in *Wallet Instance Attestation Issuance* (page 285))

- Key Attestation Issuance (detailed in *Key Attestation Issuance* (page 288)).
- Status management (maintaining and updating validity).
- Revocation processes (implementing mechanisms to revoke Wallet Instances), according to Section *Wallet Instance Revocation* (page 292).

Trust & Security Component

This component MUST ensure security through:

- Key and certificate management.
- Audit logging.
- Security monitoring and incident response.
- Compliance with IT-Wallet Federation security requirements.

Wallet Unit

User Interface

The User Interface is the point of interaction and communication between the User and the Wallet Instance.

Wallet Instance Lifecycle Management Component

Interacting with the Wallet Backend, this component MUST handle:

- Wallet Instance Registration (detailed in *Wallet Instance Initialization and Registration* (page 285)).
- Wallet Instance Attestation Issuance (detailed in *Wallet Instance Attestation Issuance* (page 285))
- Key Attestation Issuance (detailed in *Key Attestation Issuance* (page 288)).
- Status management (maintaining and updating validity).
- Revocation processes (implementing mechanisms to revoke Wallet Instances), according to Section *Wallet Instance Revocation* (page 292).

Based on the status of the Wallet Instance and the User request, this component interact with the other Wallet Instance components.

Issuer Component

Following the OpenID4VCI specification and the implementation profile in Section *Digital Credential Issuance* (page 419), this component MUST implement the Digital Credential issuance protocols and flows to request Digital Credentials to Credential Issuers.

Presentation Component

Following the implementation profile in Section *Digital Credential Presentation* (page 480), this component MUST be compliant with remote flows based on OpenID4VP and proximity flow based on ISO18013-5 .

Backup and Restore Component

For each Digital Credential that is issued to the Wallet Instance, this component MUST add all data that is necessary to request issuance of that Digital Credential during restore as specified in Section *Backup and Restore* (page 300).

Note

Currently the re-issuance of the PID and IT-Wallet ID is not managed by the Backup and Restore Component.

Dashboard and Transaction Log

This component MUST provide the Wallet Unit dashboard and transaction log functionality. In particular, this component MUST:

- provide a user-accessible interface enabling the User to access transaction transparency information;
- maintain a transaction log of transactions executed through the Wallet Unit, including non-completed transactions;
- support User interaction with transaction records, including viewing, export, and deletion.

Secure Storage

The Wallet Instance MUST use this component to protect critical assets and to securely execute cryptographic functions.

10.1.3 Wallet Solution Interaction Patterns

The Wallet Solution supports these interaction patterns:

1. **User to Wallet Backend Frontend**: Web-based interactions for Wallet Instance management.
2. **Wallet Instance to Wallet Backend API**: for Wallet Instance registration, Wallet Instance Attestation and Key Attestation issuance.
3. **PID/IT-Wallet ID Provider to Wallet Backend API**: Secure API calls to request Wallet Instance revocation.
4. **User to Wallet Instance User Interface**: for Digital Credential management (issuance, presentation, backup, restore, deletion).
5. **Wallet Instance to Relying Party**: for Digital Credential presentation.

10.1.4 Wallet Instance

The Wallet Instance establishes a strong and reliable mechanism for the User to engage in various digital transactions in a secure and privacy-preserving manner.

The Wallet Instance establishes trust with PID and (Q)EAA Providers by consistently presenting Wallet Instance and Key Attestations during interactions. These verifiable attestations, provided by the Wallet Provider, serve to authenticate the Wallet Instance itself, ensuring the trustworthiness of the WSCD, checking the Wallet Instance has not been revoked and ensuring its reliability when engaging with other ecosystem actors.

Wallet Instance Lifecycle

The Wallet Provider is in charge of the implementation and provision of Wallet Instances also handling their entire lifecycle.

In this section, state machines are presented to explain the Wallet Instance and Digital Credential states and their transitions and relations.

Note

PID is a specialized Digital Credential type that has impacts on the Wallet Instance's lifecycle. The revocation of the PID MAY also have potential impacts on (Q)EAAs, if they were issued using the presentation of the PID. When the distinction between PID and (Q)EAA is not needed, the term Digital Credential is used.

As shown in Fig. 10.2, the Wallet Instance has four distinct states: **Installed**, **Operational**, **Valid**, and **Uninstalled**. Each state represents a specific functional status and determines the actions that can be performed.

Note

The Wallet Provider MUST ensure the security and reliability of the Wallet Instances. To achieve this, the Wallet Provider MUST periodically check the Wallet Instances security and compliance status.

Transition to Installed

The state machine begins with the Wallet Instance installation (**WI INST**) transition, where Users download and install a Wallet Instance provided by the Wallet Provider using the official app store of their device's operating system (this ensures authenticity via system checks), leading to the **Installed** state.

When the state is **Installed**, the Wallet Instance MUST interact only with the Wallet Provider to be activated. When the revocation of the Wallet Instance occurs, the Wallet Instance MUST go back from **Operational** or **Valid** to **Installed**. The revocation marks the Wallet Cryptographic Hardware Key, registered during activation (see *Transition to Operational* (page 282)), as not usable anymore. Revocation can occur in the following cases:

- for technical security reasons (e.g., relating to the compromise of cryptographic material);
- in case of explicit User requests (e.g., due to loss, or theft of the Wallet Instance);
- death of the User;
- illegal activities reported by Judicial or Supervisory Bodies.

Note

As it is stated in the ARF (see Topic 9 and Topic 38 in Annex 2), the revocation of the Wallet Instance is accomplished by revoking the Wallet Instance Attestation by using the

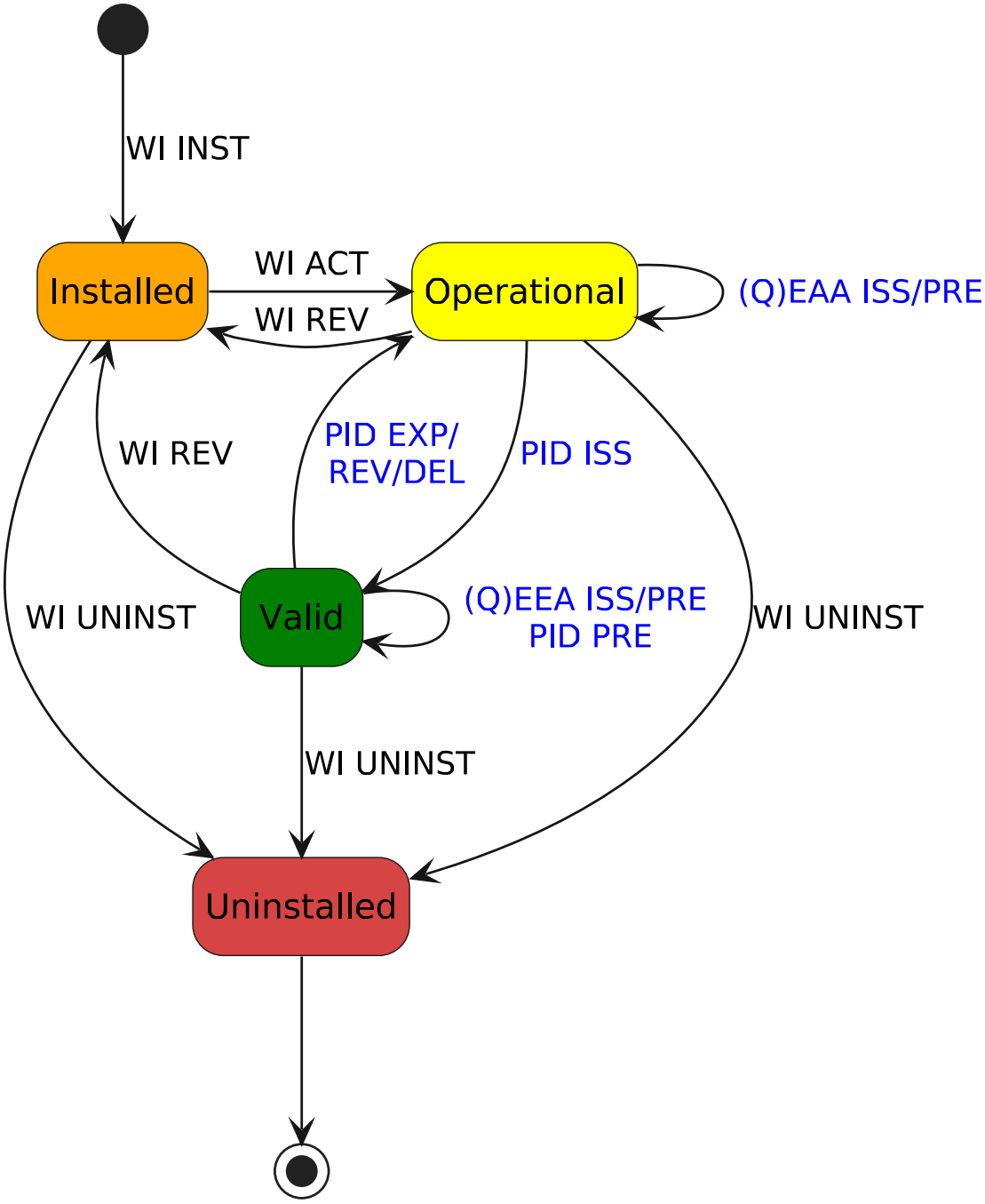


Fig. 10.2: Wallet Instance States.

status management mechanism. The Wallet Provider can simply change the status of the Wallet Instance Attestation in the Wallet Instance Attestation Status List to mark it as revoked and inform the Users concerning the reason for revoking the Wallet Instance.

Transition to Operational

After installation, the User opens the Wallet Instance and an activation begins (**WI ACT**). At this stage, a User account MUST be created with the Wallet Provider and associated with the Wallet Instance through the Wallet Cryptographic Hardware Key Tag, subject to obtaining the User's consent (see *Wallet Instance Initialization and Registration* (page 285) for more details, with tests in *WP_024* (page 739) and *WP_138* (page 767)). This association allows the User to directly request Wallet Instance revocation from the Wallet Provider.

Note

As a result of the User account creation, an authentication mechanism MUST be set for the User to interact with the Wallet Provider portal. This specification mandates the use of at least a second-factor for User authentication.

As part of the activation, the Wallet Provider MUST evaluate the operating system, general technical capabilities of the device, and trustworthiness of the WSCD to check compliance with the technical and security requirements, the authenticity and integrity of the installed Wallet Instance, and ensure the keys used for key binding resides in a secure WSCD. Upon successful verification, the Wallet Provider MUST issue at least one valid Wallet Attestation to the Wallet Instance, therefore the Wallet Instance enters the **Operational** state.

In addition, if not already done, Users MUST set their preferred method of unlocking their Wallet Instance; this MAY be accomplished by entering a personal identification number (PIN) or by utilizing biometric authentication, such as fingerprint or facial recognition, according to personal preferences and device's capabilities (*WP_025* (page 739)). Please refer to *Wallet Instance Attestation Issuance* (page 285).

In the **Operational** state, Users can request the issuance of PID (**PID ISS**), of the national **IT-Wallet ID** (which does **not** cause a transition to **Valid**), or of (Q)EAAs if the PID is not required in the issuance (**(Q)EEA ISS**). In addition, if the Digital Credentials are (Q)EEAs and for the presentation they do not require the PID, they can be presented without transitioning the Wallet Instance to another state (**(Q)EEA PRE** transition). The IT-Wallet ID MAY likewise be presented while the Wallet Instance remains **Operational**.

A **Valid** Wallet Instance MUST transition back to the **Operational** state due to **PID EXP/REV/DEL** transition, when the associated PID expires, is revoked by its Provider or either deleted by the User.

Transition to Valid

A transition to the Valid state occurs only when the Wallet Instance obtains a valid PID (**PID ISS**). Obtaining an IT-Wallet ID MUST NOT cause this transition. In the **Valid** state, Users can obtain and present new (Q)EAAs (**(Q)EAA ISS/PRE**), and present the PID (**PID PRE**). Please refer to *Digital Credential Issuance* (page 419) and *Digital Credential Presentation* (page 480).

Note

Users can have only one Wallet Instance in **Valid** state for the same Wallet Solution. Thus, when a User installs and obtains a PID on a new Wallet Instance of the same Wallet Solution from the same Wallet Provider, the PID in the previous Wallet Instance MUST be revoked and the Wallet Instance became **Operational**.

Transition to Uninstalled

Across all states, **Installed**, **Activated**, **Operational**, or **Valid**, the Wallet Instance can be removed entirely through the Wallet Instance uninstall (**WI UNINST**) transition, leading to the **Uninstalled** state. If a Wallet Instance is **Uninstalled** it ends its lifecycle (*WP_139* (page 767)).

Wallet Instance Lifecycle Management

While Fig. 10.2 shows the different states a Wallet Instance may acquire during its lifecycle, Fig. 10.3 shows the point of view of Wallet Instances and Wallet Providers in managing the Wallet Instance lifecycle and the effect on their local storage.

Through a Wallet Instance in an **Installed** state, a User is able to start the **Wallet Instance Activation** (**WI ACT**). As a result, the Wallet Instance MUST create a Wallet Cryptographic Hardware Key pair. In addition, if not already done, Users MUST set their preferred method of unlocking their Wallet Instance. As a result of the **Wallet Instance Revocation** (**WI REV**), the Wallet Instance MUST delete the Wallet Cryptographic Hardware Key pairs (*WP_149* (page 767)).

A Wallet Provider instead is responsible for:

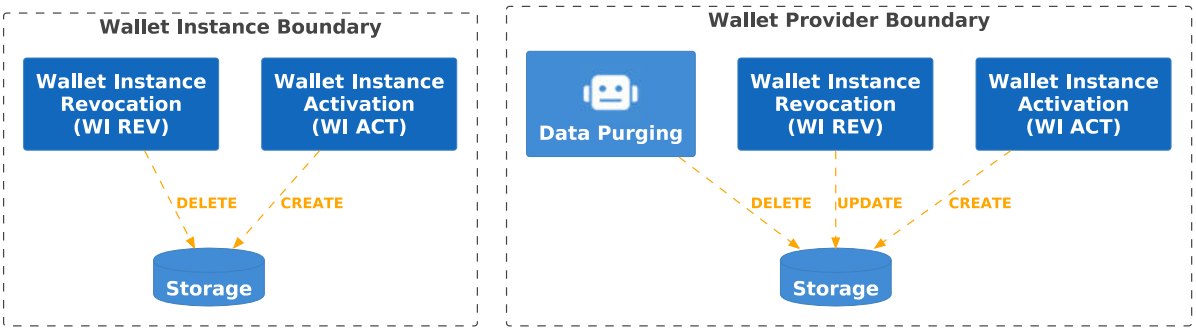


Fig. 10.3: Wallet Instance Lifecycle Management.

- **Wallet Instance Activation** (**WI ACT**): a User account MUST be created and associated with the Wallet Instance through the Wallet Cryptographic Hardware Key Tag. As a result of the User account creation, an authentication mechanism of at least two factors MUST be set for the User to interact with the Wallet Provider portal.
- **Wallet Instance Revocation** (**WI REV**): for technical security reasons or triggered by external entities (e.g., Users and Supervisory Bodies) the Wallet Instance is revoked and this result in the revocation of the Wallet Instance Attestation, which is reflected on the Wallet Instance Attestation Status List. Additionally, the Wallet Cryptographic Hardware Key Tag MUST be deleted from the User account.
- **Data Purging**: through an explicit request of Users, the User account at the Wallet Provider MUST be removed from the local storage.

Wallet Instance Functionalities

A Wallet Instance, MUST support the following functionalities:

- Wallet Registration (detailed in *Wallet Instance Initialization and Registration* (page 285)),
- Wallet Instance Attestation Issuance (detailed in *Wallet Instance Attestation Issuance* (page 285)),
- Key Attestation Issuance (detailed in *Key Attestation Issuance* (page 288)),
- Wallet Revocation (detailed in *Wallet Instance Revocation* (page 292)) and
- Deletion of presented attributes (detailed in *User's Attributes Deletion* (page 297)).

Each functionality is described in detail in the following sections.

Note

The details provided below are non-normative and are intended to clarify the functionalities of the Wallet Instance Registration. The actual implementation may vary based on the specific use case and requirements of the Wallet Provider.

Wallet Instance Initialization and Registration

This process allows the User who has just installed the Wallet Instance application to register the Wallet Instance with the Wallet Provider Backend. During this process, the Wallet Instance application will request a security and integrity assertion from the OS manufacturer, which also binds a long-lived key pair stored in a proper secure storage within the device itself. This assertion will be validated by the Wallet Provider, and if the validation is successful, the Wallet Provider will authenticate the Wallet Instance. For details see *Mobile Application Instance Initialization* (page 667).

Warning

During the registration phase of the Wallet Instance with the Wallet Provider it is also necessary to associate the Wallet Instance with a specific User, authenticating the User with the Wallet Provider. The authentication mechanism is at the discretion of the Wallet Provider and it will not be addressed within these guidelines, as each Wallet Provider may have its User authentication systems already implemented.

Note

The Wallet Provider SHOULD associate the Wallet Instance (through the `hardware_key_tag` identifier) with a specific User uniquely identified within the Wallet Provider's systems. This will be useful for the lifecycle of the Wallet Instance and for a future revocation. For details see *Mobile Application Instance* (page 667).

Wallet Instance Attestation Issuance

This section describes how the Wallet Provider issues a Wallet Instance Attestation.

Step 1: The User initiates a new operation that necessitates the acquisition of a Wallet Instance Attestation.

Steps 2-3: The Wallet Instance MUST:

1. Verify the existence of Cryptographic Hardware Keys. If none exist, Wallet Instance re-initialization is required (*WP_140a* (page 767)).

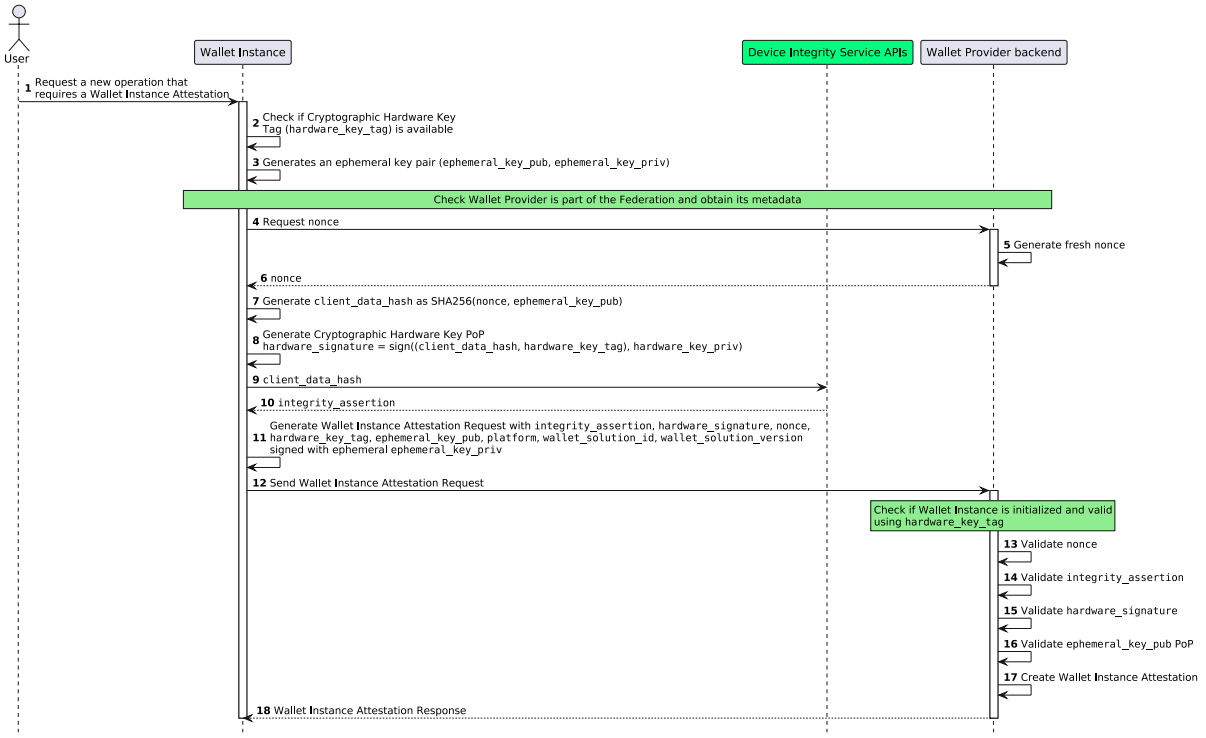


Fig. 10.4: Sequence Diagram for Wallet Instance Attestation acquisition.

2. Generate an ephemeral asymmetric key pair for Wallet Instance Attestation (`ephemeral_key_pub`, `ephemeral_key_priv`), linking the public key to the attestation (*WP_026* (page 739)).
3. Verify the Wallet Provider's federation membership and retrieve its metadata (*WP_023* (page 739)).

Steps 4-6 (Nonce Retrieval): The Wallet Instance requests a `nonce` value from the *Wallet Solution Nonce Endpoint* (page 544) of the Wallet Provider Backend (*WP_140b* (page 767)). The `nonce` is required to be unpredictable and serves as the main defense against replay attacks.

The `nonce` MUST ensure single-use within a predetermined time frame.

Upon a successful request, the Wallet Provider generates and returns the nonce value to the Wallet Instance.

Step 7: The Wallet Instance performs the following actions (*WP_140c* (page 767)):

- Creates `client_data`, a JSON object that includes the `nonce` and the thumbprint of `ephemeral_key_pub` JWK.
- Computes `client_data_hash` by applying the `SHA256` algorithm to the `client_data`.

Below is a non-normative example of the `client_data` JSON object.

```
{
  "nonce": "i4ThI2Jhbu81i8mqyWEuDG5t",
```

```
  "jwk_thumbprint": "vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"
}
```

Steps 8-10: The Wallet Instance:

- produces an `hardware_signature` value by signing the `client_data_hash` with the Wallet Hardware's private key, serving as a proof of possession for the Cryptographic Hardware Keys (*WP_140d* (page 767)).
- requests the Device Integrity Service to create an `integrity_assertion` value for Wallet Instance Attestation linked to the `client_data_hash`.
- receives a signed `integrity_assertion` value for Wallet Instance Attestation from the Device Integrity Service, authenticated by the OEM (*WP_140e* (page 767)).

Note

`integrity_assertion` is a custom payload generated by Device Integrity Service, signed by device OEM and encoded in base64 to have uniformity between different devices.

Steps 11-12 (Wallet Instance Attestation Issuance Request): The Wallet Instance:

- Constructs the Wallet Instance Attestation Request in the form of a JWT. This JWT includes the `integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, `platform`, `wallet_solution_id`, `wallet_solution_version` and other configuration related parameters (see *Table of the Wallet Instance Attestation Request Body* (page 548)) and is signed using the private key of the initially generated ephemeral key pair (*WP_140–141* (page 767)).
- Submits the Wallet Instance Attestation Request to the *Wallet Instance Attestation Issuance Endpoint* (page 547) of the Wallet Provider Backend.

The Wallet Instance MUST send the signed Wallet Instance Attestation Request JWT as an `assertion` parameter in the body of an HTTP request to the Wallet Provider's *Wallet Instance Attestation Issuance Endpoint* (page 547) (*WP_142* (page 767)).

Steps 13-17: The Wallet Provider Backend evaluates the Wallet Instance Attestation Request and MUST perform the following checks (*WP_143* (page 767)):

1. The request MUST include all required HTTP header parameters as defined in *Wallet Instance Attestation Issuance Request* (page 547) (*WP_143a* (page 767)).
2. The signature of the Wallet Instance Attestation Request MUST be valid and verifiable using the provided `jwk` (*WP_143b* (page 767)).
3. The `nonce` value MUST have been generated by the Wallet Provider and not previously used (*WP_143c* (page 767)).

4. A valid and currently registered Wallet Instance associated with the provided `hardware_key_tag` MUST exist (*WP_143d* (page 767)).
5. The `client_data` MUST be reconstructed using the `nonce` and the `jwk` public key. The `hardware_signature` parameter value is then validated using the registered Cryptographic Hardware Key's public key associated with the Wallet Instance (*WP_143e* (page 767)).
6. The `integrity_assertion` MUST be validated according to the device manufacturer's guidelines. The specific checks performed by the Wallet Provider are detailed in the operating system manufacturer's documentation (*WP_143f* (page 767)).
7. The device in use MUST be free of known security flaws and meet the minimum security requirements defined by the Wallet Provider.
8. The URL in the `iss` parameter MUST match the Wallet Provider's URL identifier (*WP_143g* (page 767)).

Upon successful completion of all checks, the Wallet Provider issues a Wallet Instance Attestation that meets *the Wallet Instance Attestation lifetime* (page 272) (*WP_028* (page 739), *WP_144* (page 767)).

Step 18 (Wallet Instance Attestation Issuance Response): Upon successful completion, the Wallet Provider MUST return a confirmation response using status code 200 and Content-Type `application/json`, containing the Wallet Instance Attestation signed by the Wallet Provider. The Wallet provider MUST return the Wallet Instance Attestation in JWT format. The Wallet Instance will then perform security and integrity verification of the Wallet Instance Attestation received in addition to trust verification of its Issuer (*WP_030–031* (page 739)).

Below is a non-normative example of the response.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "wallet_instance_attestation": "omppc3N1ZXJBdXRohEOhASaiBE...dElEAnFlbGVtZW50SWRl"
}
```

Key Attestation Issuance

This section describes how the Wallet Provider issues Key Attestations.

Step 1: The User initiates a new operation that necessitates the acquisition of a Key Attestation.

Steps 2-3: The Wallet Instance MUST:

1. Verify the existence of Cryptographic Hardware Keys. If none exist, Wallet Instance re-initialization is required (*WP_140a* (page 767)).

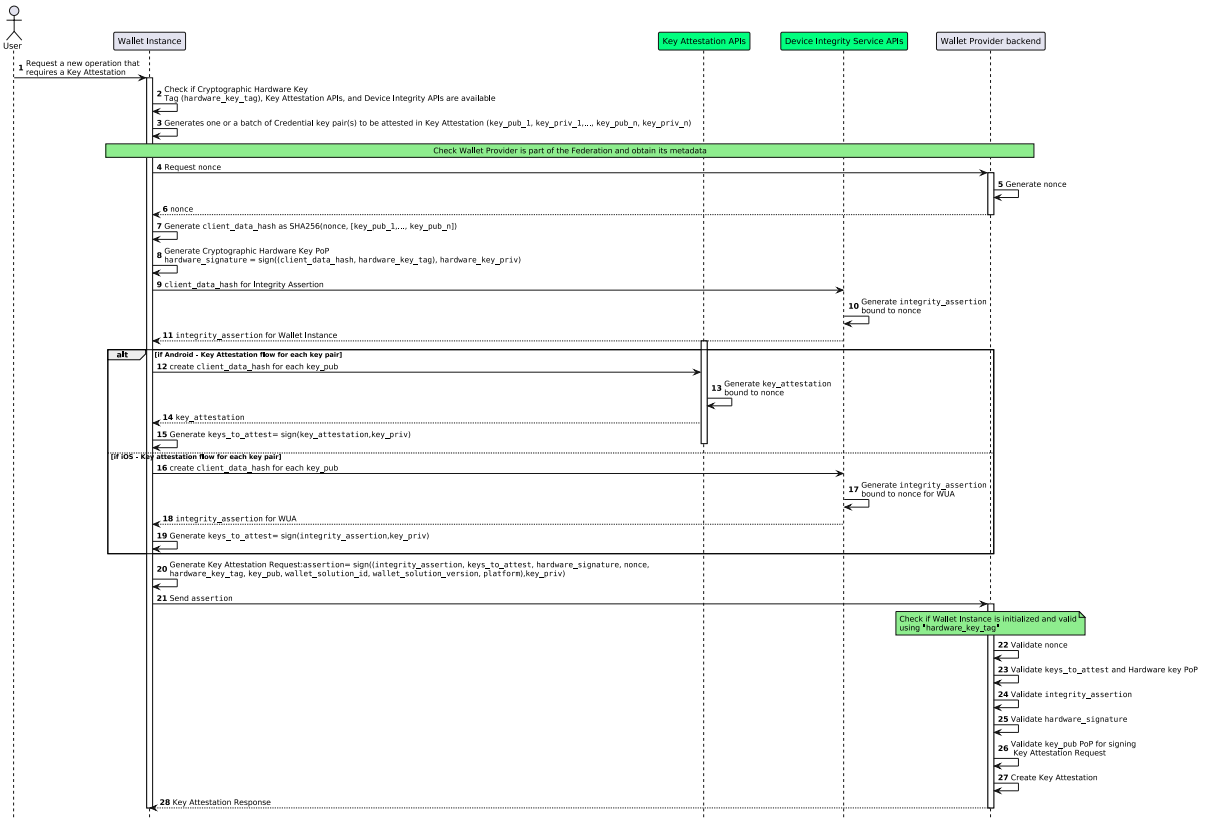


Fig. 10.5: Sequence Diagram for Key Attestations acquisition.

2. Generate one or batch of asymmetric Credential key pair(s) to be attested in Key Attestation (`key_pub_1`, `key_priv_1`, ..., `key_pub_n`, `key_priv_n`).
3. Verify the Wallet Provider's federation membership and retrieve its metadata (*WP_023* (page 739)).

Steps 4-6 (Nonce Retrieval): The Wallet Instance requests a `nonce` value from the *Wallet Solution Nonce Endpoint* (page 544) of the Wallet Provider Backend (*WP_140b* (page 767)). The `nonce` is required to be unpredictable and serves as the main defense against replay attacks.

The `nonce` MUST ensure single-use within a predetermined time frame.

Upon a successful request, the Wallet Provider generates and returns the nonce value to the Wallet Instance.

Step 7: The Wallet Instance performs the following actions (*WP_140c* (page 767)):

- Creates `client_data`, a JSON object that includes the `nonce` and a `jwk_thumbprints` field containing a JSON array of the JWK thumbprints corresponding to the public keys `(key_pub_1,...,key_pub_n)` .
- Computes `client_data_hash` by applying the `SHA256` algorithm to the `client_data`.

Below is a non-normative example of the `client_data` JSON object.

```
{
  "nonce": "i4ThI2Jhbu81i8mqyWEuDG5t",
  "jwk_thumbprints": ["vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"]
}
```

Steps 8-11: The Wallet Instance:

- produces an `hardware_signature` value by signing the `client_data_hash` with the Wallet Hardware's private key, serving as a proof of possession for the Cryptographic Hardware Keys (*WP_140d* (page 767)).
- requests the Device Integrity Service to create an `integrity_assertion` value linked to the `client_data_hash`.
- receives a signed `integrity_assertion` value for Wallet Instance from the Device Integrity Service, authenticated by the OEM (*WP_140e* (page 767)).

Note

`integrity_assertion` is a custom payload generated by Device Integrity Service, signed by device OEM and encoded in base64 to have uniformity between different devices.

Note

In the case of Android OS, the flow continues based on **Steps 12-15**, otherwise for the case of iOS, the flow continues based on **Steps 16-19**.

Steps 12-15: The Wallet Instance:

- requests the Key Attestation API to create an `key_attestation` value for each `client_data_hash` per each `key_pub`.
- receives a signed `key_attestation` value from the Key Attestation API, authenticated by the OEM.
- generate `keys_to_attest` value by signing the `key_attestation` using the private key of the initially generated credential key pair (`priv_key`).

Steps 16-19: The Wallet Instance:

- requests the Device Integrity Service to create an `integrity_assertion` value for each `client_data_hash` per each `key_pub`.
- receives a signed `integrity_assertion` value from the Device Integrity Service, authenticated by the OEM.

- generate `keys_to_attest` value by signing the `integrity_assertion` that is obtained for the Key Attestation using the private key of the initially generated credential key pair (`priv_key`).

Steps 20-21 (Key Attestation Issuance Request): The Wallet Instance:

- Constructs the Key Attestation Request in the form of a JWT. This JWT includes the `integrity_assertion`, `keys_to_attest`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, `platform` and other configuration related parameters (see *Table of the Key Attestation Request Body* (page 554)). The Key Attestation Request MUST be signed using the private key related to the public key included in the request, using the `cnf` parameter (first element of `keys_to_attest`) (*WP_140–141* (page 767)).
- Submits the Key Attestation Request to the *Key Attestation Issuance Endpoint* (page 554) of the Wallet Provider Backend.

Note

`keys_to_attest` contains a `key_attestation` object in case of Android and `integrity_assertion` in case of iOS.

The Wallet Instance MUST send the signed Key Attestation Request JWT as an `assertion` parameter in the body of an HTTP request to the Wallet Provider's *Key Attestation Issuance Endpoint* (page 554) (*WP_142* (page 767)).

Steps 22-27: The Wallet Provider Backend evaluates the Key Attestation Request and MUST perform the following checks (*WP_143* (page 767)):

1. The request MUST include all required HTTP header parameters as defined in *Key Attestation Issuance Request* (page 554) (*WP_143a* (page 767)).
2. The signature of the Key Attestation Request MUST be valid and verifiable using the provided `jwk` (*WP_143b* (page 767)).
3. The `nonce` value MUST have been generated by the Wallet Provider and not previously used (*WP_143c* (page 767)).
4. A valid and currently registered Wallet Instance associated with the provided `hardware_key_tag` MUST exist (*WP_143d* (page 767)).
5. The signature of the `keys_to_attest` parameter Must be first validated using the provided `jwk` and its value (`key_attestation` in case of Android or `integrity_assertion` in case of iOS) MUST be validated according to the device manufacturer's guidelines.

6. The `client_data` MUST be reconstructed using the `nonce` and their respective thumbprint JWKs `[key_pub_1,...,key_pub_n]`. The `hardware_signature` parameter value is then validated using the registered Cryptographic Hardware Key's public key associated with the Wallet Instance (*WP_143e* (page 767)).
7. The `integrity_assertion` MUST be validated according to the device manufacturer's guidelines. The specific checks performed by the Wallet Provider are detailed in the operating system manufacturer's documentation (*WP_143f* (page 767)).
8. The device in use MUST be free of known security flaws and meet the minimum security requirements defined by the Wallet Provider.
9. The URL in the `iss` parameter MUST match the Wallet Provider's URL identifier (*WP_143g* (page 767)).

Upon successful completion of all checks, the Wallet Provider issues a Key Attestation valid for at least one month.

Step 28 (Key Attestation Issuance Response): Upon successful completion, the Wallet Provider MUST return a confirmation response using status code set with `200` and Content-Type `application/json`. The response MUST contain the Key Attestations signed by the Wallet Provider using the JWT format. The Wallet Instance MUST perform security and integrity verification of the Key Attestations received, along with the trust verification already evaluated about the Issuer (*WP_030–031* (page 739)).

Below is a non-normative example of the response.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "key_attestation": "omppc3N1ZXJBdXRohEOhASaiBE...dElEAnFlbGVtZW50SWRl"
}
```

Wallet Instance Revocation

This section describes the involved entities and modalities to request a Wallet Instance revocation.

The Wallet Provider MUST ensure the security and reliability of Wallet Instances, keeping them updated and compliant with security requirements. When, for technical security reasons (e.g. , relating to the compromise of cryptographic material) the security of the Wallet Instance is compromised, the Wallet Provider MUST revoke the Wallet Instance.

As shown in Fig. 10.6, other actors MAY trigger the Wallet Instance revocation process (*WP_007–009* (page 737)):

- **Users**, connecting to the Wallet Provider's web portal from their Wallet Instance or using an external browser.
- **PID Providers and IT-Wallet ID EAA Providers** when notified by the Authentic Source of the PID/IT-Wallet ID (ANPR) of the User's death.
- **Legal Authorities or the Supervisory Body** in cases of proven illegal activities.

Regardless of who triggered the revocation, the Wallet Provider MUST inform Users when their Wallet Unit is revoked, according to the following requirements (*WP_034* (page 739)):

- Out-of-band alerts (e.g., email or SMS) MUST be sent as the primary notification mechanism alongside the Wallet Unit itself.
- The alert MUST occur within 24 hours from the effective Wallet Unit revocation.
- A user-friendly explanation about the reason of the revocation MUST be provided to the User alongside with the instruction to reactive the Wallet Unit if possible.

Note

- The flow for the Wallet Instance Revocation triggered by the User is detailed below.
- The endpoint used by the PID/IT-Wallet ID Provider is detailed in the Wallet Provider Catalog of e-Service PDND Catalog (see Section *e-Service PDND Wallet Provider Catalog* (page 561) for technical details).
- The flow for Authorized Entities (e.g., Supervisory Bodies) is out of scope of this specification, it will be managed by each Wallet Provider.

Wallet Instance Revocation Request

Users MAY request the Wallet Instance revocation (*WP_032* (page 739)) by:

- *Selecting the revocation functionality from their Wallet Instance*: this functionality may be used by Users before changing their phone.
- *Using an external user agent*: this covers cases where Users lose their device, and so their access to their Wallet Instance.

In both cases, by using the Wallet Provider portal (*WP_005–006* (page 737)):

- Users MUST authenticate with at least a second-factor authentication mechanism, or have an active session that meets this requirement.
- The Wallet Provider MUST allow Users to view the state of their Wallet Instances associated with their authenticated session and ask for revocation, sending a Wallet Instance

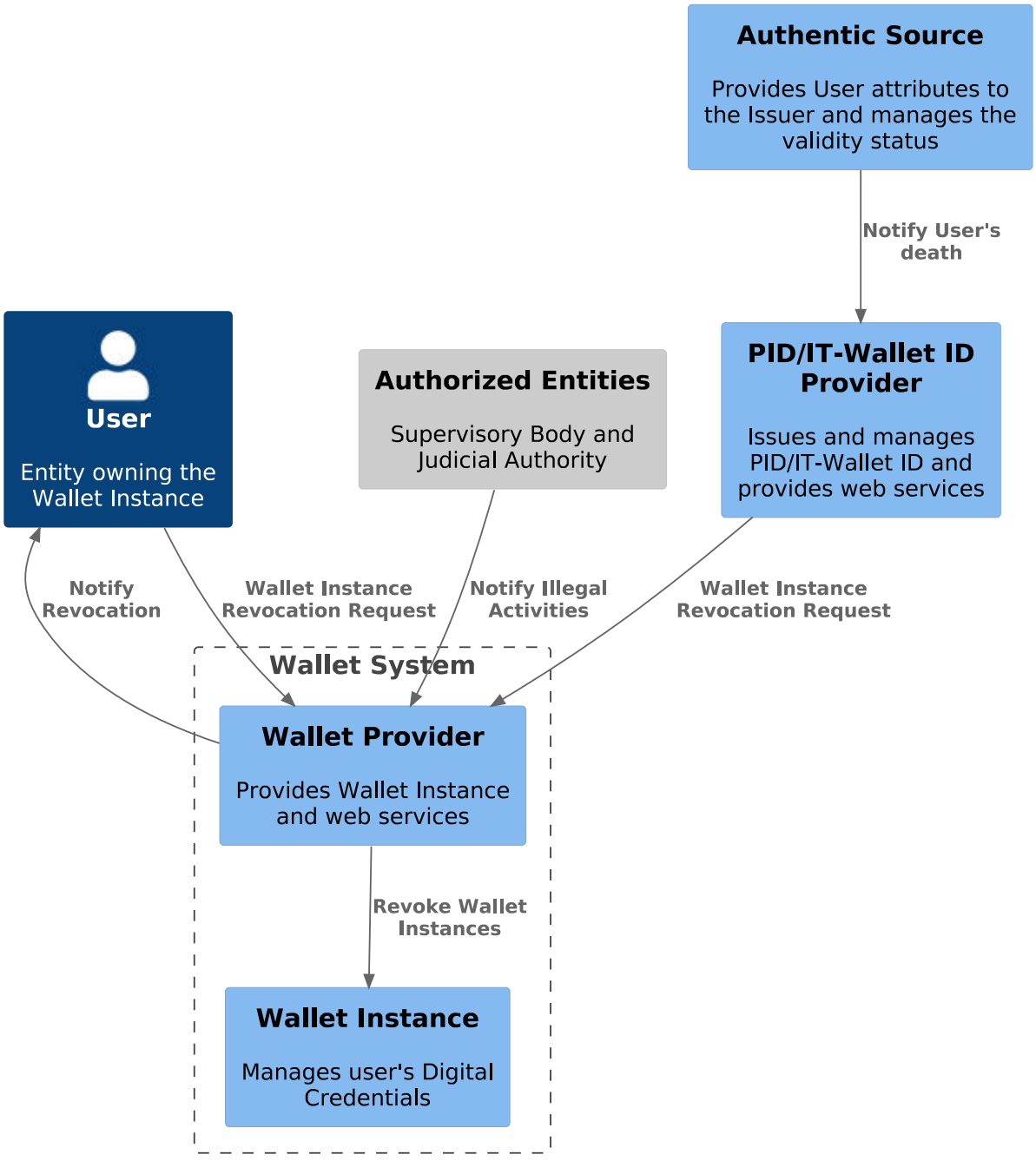


Fig. 10.6: Entities involved in the Wallet Instance revocation process.

Retrieval or Revocation Request (*WP_033* (page 739), and *WP_145* (page 767)), as applicable, to the *Wallet Instance Management Endpoint* (page 544) of the Wallet Provider Backend.

Below is a non-normative example of a Wallet Instances Retrieval Request.

```
GET /wallet-instances HTTP/1.1
Host: walletprovider.example.com
```

Upon a successful retrieval, the Wallet Provider MUST return a confirmation response, with the status of all Wallet Instances associated with the User (*WP_146* (page 767)). Below is a non-normative example of a Wallet Instances Retrieval Response.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

[
  {
    "id": "f7b2a8d9",
    "status": "ACTIVE",
    "issued_at": "2024-03-12T10:00:00Z"
  },
  {
    "id": "g8b235c4",
    "status": "REVOKED",
    "issued_at": "2024-02-28T15:30:00Z"
  }
]
```

Once the User identifies the Wallet Instance to be revoked, a Wallet Instance Revocation Request can be sent to the endpoint, including the Wallet Instance ID as a path parameter (*WP_147* (page 767)). Below is a non-normative example of a Wallet Instance Revocation Request.

```
PATCH /wallet-instances/{f7b2a8d9} HTTP/1.1
Host: wallet-provider.example.org
Content-Type: application/json

{
  "status": "REVOKED"
}
```

Wallet Instance Revocation Response

Upon a successful revocation, the Wallet Provider MUST return a confirmation response (*WP_148* (page 767)). Below is a non-normative example of a Wallet Instance Revocation Response.

```
HTTP/1.1 204 No Content
```

Revocation Check Mechanisms

The verification of the Wallet Instance validity MUST be performed:

- **During Digital Credential issuance** by the Credential Issuers. Only Wallet Instances in Operational or Valid state have valid Wallet Instance Attestation and Key Attestation. The verification of the validity of a Wallet Instance is indirectly performed by Credential Issuers by checking the presence of a valid Wallet Instance Attestation (i.e., not expired, not revoked checking the Wallet Instance Attestation Status List, and signed by a trusted Wallet Provider).
- **During the validity period of the Digital Credential** by the Credential Issuers every 24 hours by checking the Wallet Instance Attestation Status List of the Wallet Instance holding the PID, and the Key Attestation Status List. Indeed, if the Wallet Instance or the Key Attestation is revoked, the PID hosted within it MUST be revoked. Any other Digital Credential obtained through the presentation of the PID MUST therefore be revoked too.
- **During the Wallet Instance lifecycle** by the Wallet Instance. Each Wallet Provider may implement different methods to allow a Wallet Instance to check its status. The IT Wallet specification RECOMMENDS to use the Status List of the Wallet Instance Attestation.

Note

In the current version of the specification, Credential Issuers are directly notified of a Wallet Instance revocation by the Wallet Provider using a PDND e-service.

Note

During the Digital Credential presentation phase, a Relying Party can indirectly check the Wallet Instance revocation by checking the Digital Credential revocation. When a Wallet Instance is revoked, the Wallet Provider revokes the corresponding Wallet Instance Attestations, triggering the revocation of the Digital Credential by Credential Issuer.

User's Attributes Deletion

This Wallet Instance functionality allows Users to obtain a list of all Relying Parties towards which Digital Credentials have been presented. Subsequently Users may request deletion of the attributes of one or more Digital Credentials presented to a Relying Party of their choice. The Relying Party MUST uniquely identify those Digital Credentials by applying identity matching as specified in *Identity Matching and Identity Reconciliation* (page 538). Below the high level flow regarding this interaction is presented (*WP_115* (page 764)).

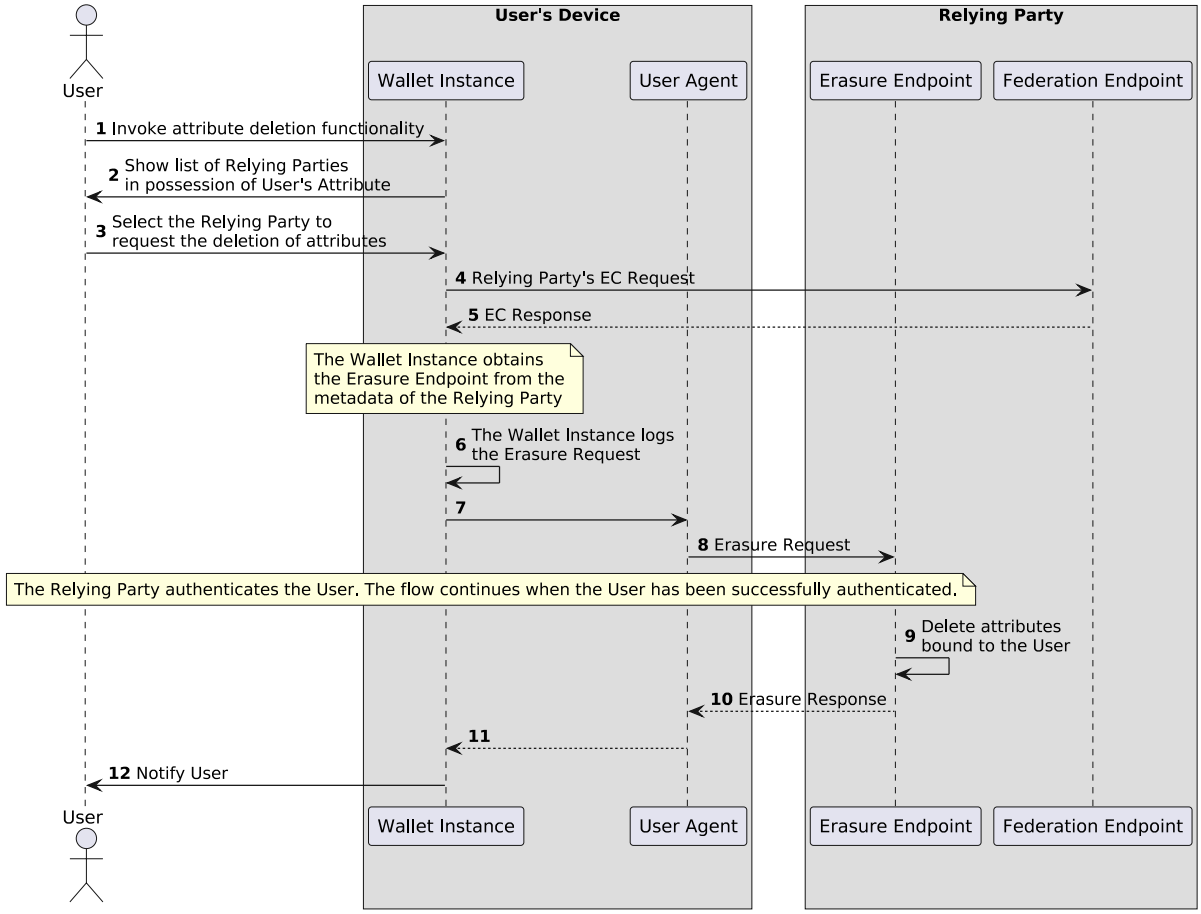


Fig. 10.7: Deletion of User's Attributes Process.

Step 1: The User requests the deletion of attributes invoking the Wallet Instance's attribute deletion function.

Step 2: The Wallet Instance collects all transaction data and shows the User the list of Relying Parties with which it has had interactions throughout the Wallet Instance lifecycle and are in possession of User's attributes (*WP_115a* (page 764)).

Step 3: The User selects the target Relying Party for attributes deletion.

Steps 4 - 5: The Wallet Instance obtains the Relying Party Entity Configuration at the Federation ./well-known/ endpoint. The URL or the Erasure Endpoint (`erasure_endpoint`) can be found inside the metadata parameter (*WP_116* (page 764)).

Step 6: The Wallet Instance logs the Erasure Request's relevant information as defined in *Wallet Instance Dashboard and Transaction Logging* (page 299). These logs MUST include at least (*WP_117a* (page 764)):

- the date of request,
- the Relying Party to which the request was made,
- the attributes requested to be removed.

Steps 7 - 8: The Wallet Instance redirects the User to the Erasure Endpoint. It MUST also ensure that a callback mechanism to allow the User-Agent to notify the Wallet Instance (and thus the User) after the Erasure Response is present (*WP_118* (page 764)). Details on the Erasure Request can be found in relying-party-provider-backend-endpoint:Relying Party Provider Backend Erasure Endpoint (*WP_117* (page 764)).

Note

The Relying Party web page will authenticate the User with an appropriate Level of Assurance using any method such as CIE or the PID/IT-Wallet ID presentation. The specific mechanism used for authentication is left to the Relying Party. If the User does not authenticate through the Wallet Instance, identity matching and identity reconciliation MAY use a preexisting national authentication scheme, where possible, as specified in *Identity Matching and Identity Reconciliation* (page 538). Upon authenticating the User, the Relying Party MAY prompt the User to perform additional steps needed for the deletion of attributes, e.g. , it might require the User to confirm the deletion operation.

Step 9: Upon successful authentication of the User, the Relying Party MUST uniquely identify one or more Digital Credentials for which the User requests deletion by applying identity matching, and MUST delete the related attributes in its possession.

Step 10: The Relying Party returns the Erasure Response in the form of an HTTP Response to the User-Agent and includes the callback URL if provided in the Erasure Request. Details on the Erasure Response can be found in relying-party-provider-backend-endpoint:Relying Party Provider Backend Erasure Endpoint.

Steps 11 - 12: The User-Agent uses the implemented method to return the Erasure Response to the Wallet Instance. Finally, the User is notified via the Wallet Instance regarding the Erasure Response outcome (*WP_119* (page 764)).

Wallet Instance Dashboard and Transaction Logging

The Wallet Instance MUST maintain a secure, Wallet-local transaction log to ensure transparency, traceability, and User awareness of actions performed through the Wallet Instance as defined in EIDAS-ARF (Annex 2.02 HLRs, Topic 19). The log MUST cover all transactions (including non-completed) in these categories:

- credential issuance and re-issuance;
- credential presentation to Relying Parties;
- data deletion requests sent to Relying Parties.

For each such transaction, the Wallet Instance MUST create a corresponding transaction record with the data elements specified in the applicable ARF requirements for transaction logging for the corresponding transaction category, and update it with the final outcome where applicable as defined in EUDI-TS 10.

For presentation transactions, attribute values MUST NOT be logged. If a presentation request contains sensitive transactional data (such as a payment or a digital signature), the Wallet Instance MUST log the value of such transactional data only to the extent explicitly required by EUDI-TS 12 associated with the requested attestation and in accordance with data minimization principles. Where no such requirement exists, transactional data values MUST NOT be logged.

To support User access to the transaction log, the Wallet Instance MUST provide a dashboard interface, allowing the User to view an overview of transaction records in the log, access individual record details, export and delete records, and, where a transaction involves a Relying Party, initiate a data deletion request to the corresponding Relying Party (using logged contact information, if available).

The Wallet Provider MUST protect the confidentiality, integrity, and authenticity of the transaction log and of any exported log objects in accordance with EUDI-TS 10; access to the log MUST be restricted to the User and the Wallet Instance.

Records MUST be retained for at least the minimum legal period as defined in *General Log Retention Policies* (page 635). If storage limits require automatic deletion, the Wallet Instance MUST notify the User via the dashboard, warn of consequences, and offer export of affected records before deletion.

Export and Deletion of Transaction Records

Export: The dashboard MUST allow the User to export one or more transaction records to a file using the common format and security measures defined in EUDI-TS 10.

The exported file MUST be storable in an external or remote storage location of the User's choice, from among the storage options supported by the Wallet Instance.

Deletion: The dashboard MUST allow the User to delete one or more transaction records from the Wallet Instance at any time.

Before deleting any transaction record, the Wallet Instance MUST display a clear warning to the User about the potential consequences for the User's data protection rights.

Deletion of transaction records MUST:

- apply only to Wallet-local copies of the records;
- be irreversible for the Wallet Instance once completed.

The Wallet Instance MUST continue to protect any remaining transaction records in accordance with the confidentiality, integrity, and authenticity requirements defined for the transaction log in this section.

No entity other than the User MAY delete transaction records.

10.1.5 Backup and Restore

The **Backup and Restore** functionality becomes relevant when the *User* wants to change device and/or *Wallet Solution* in order to access the previously issued Digital Credentials again.

- The mobile device has been **lost**, **stolen**, **damaged**, or **compromised** (e.g., due to unauthorized access).
- The *User* replaces an existing *Wallet Instance* with a new instance of the same *Wallet Solution*.
- The *User* has changed their mobile device and needs to set up the *Wallet Solution* on the new device.
- The *User* performs a factory reset on the current phone and needs to set up the *Wallet Solution* again.

Note

For Wallet Solutions based on the IT Wallet *Technical Specifications*, the migration to a different *Wallet Solution* (known as data portability) can be supported following the backup and restore functionality described in this section.

Backup Flow

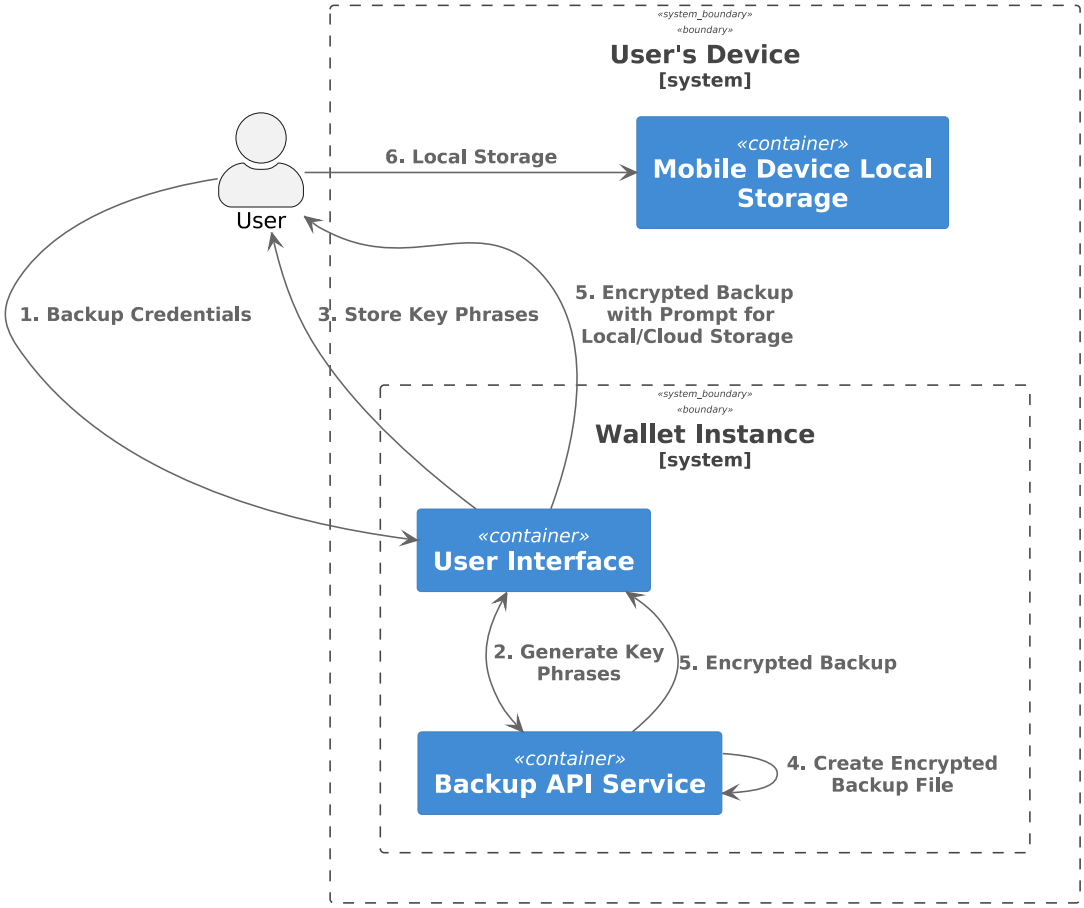


Fig. 10.8: Backup flow

Below, the description of the steps of Fig. 10.8:

Step 1: The *User* selects the option to back up Credentials stored within the *Wallet Instance* (*WP_120* (page 765)).

Steps 2-3: The *Wallet Instance* using the backup APIs randomly selects 10 key phrases from a pre-generated list of words and displays it to the User (*WP_120a* (page 765)). The *User* MUST securely store the key phrase chosen from those proposed by the system (e.g., in a password manager or a physical safe) as they are critical for restoring the backup (*WP_120b* (page 765)).

Note

As highlighted in the ARF, encryption is necessary because the backup file is considered sensitive. Even if an attacker only knows the *Issuer* identifiers, they can infer the different types of Digital Credentials, which constitutes a violation of *User* privacy.

Note

To extract the key from the list of selected words a key derivation function MUST be applied. Password-Based-Key-Derivation Function 2 (PBKDF2) is among the mostly used ones based on RFC 2898 and it is recommended by the NIST 800-132. There are other relevant techniques that are available and used widely, such as Bcrypt, Scrypt, and Argon2. More details on this approach can be found here (*WP_121* (page 765)).

Note

The work factor for PBKDF2 is implemented through an iteration count, which should set differently based on the internal hashing algorithm used. The recommended value for `SHA-256` hashing algorithm is 600000 iterations as stated in the OWASP Password Storage Cheatsheet (*WP_121a* (page 765)).

Step 4: The *Wallet Instance* performs the operations below to create the backup JWT entry for the backup file (*WP_122* (page 765)).

- For each of the HW bound key Credentials, add the *Credential Issuer* identifier and the `credential_configuration_id` as an entry in the backup JWT (*WP_122d* (page 765)).
- Sign the backup JWT using the private key that its public key is attested within the *Wallet Instance Attestation*. The related public key that is attested by the *Wallet Provider* is provided within the *Wallet Instance Attestation* (`cnf` claim). The *Wallet Instance* MUST verify the validity of the *Wallet Instance Attestation* before signing the backup JWT (*WP_123* (page 765)).
- Add the signed backup JWT as an entry to the backup file (*WP_122* (page 765)).
- Encrypt the backup file using the provided key phrase (*WP_124* (page 765)).

Note

The Backup JWT MAY contain transaction history for each Credential entry within the `credentials_backup` claim (*WP_122c* (page 765)).

Step 5: The *User* will be prompted to choose a storage option for securely storing the backup file. Options may include native storage or external storage solutions, such as cloud storage, USB devices, e-mail delivery or any other (*WP_125* (page 765)).

Step 6: In the case where the *User* prefers the native storage, the backup file is stored on the *User* device.

A non-normative example of the backup JWT header and payload is as the following:

```
{
  "alg": "ES256",
  "typ": "wallet-unit-credentials-backup+jwt"
}
```

```
{
  "timestamp":"2024-12-13T16:35:06+01:00",
  "wallet_provider_id":"https://wallet-provider.example.org/",
  "wallet_instance_version":"v1.0",
  "wallet_instance_attestation":"eyJhbGciOiJFUzI1NiIsImVVfQz.eyJpc3MiOiAiaH...LCAibmJ",
  "credentials_backup": {
      "https://issuer.example.org/v1.0/mDL": ["org.iso.18013-5.1.mDL"],
      "https://eaa-provider.example.org/": ["dc_sd_jwt_EuropeanDisabilityCard"]
   }
}
```

The JOSE header of the backup JWT MUST contain the following REQUIRED parameters (*WP_122a* (page 765)):

JOSE header	Description	Reference
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	**RFC 7516# section-4.1.1**.
typ	It MUST be set to `wallet-unit-credentials-backup+jwt`	N/A

The body of backup JWT contains the following REQUIRED claims (*WP_122b* (page 765)):

Claim	Description
timestamp	UNIX timestamp with the time of backup file creation. This value is updated every time a new Credential entry is added to the backup file.
wallet_provider_id	It MUST be set to the unique identifier of the *Wallet Provider*.
wallet_instance_version	It MUST be set to the version of the *Wallet Solution* that has been backed up.

continues on next page

Table 10.2 – continued from previous page

Claim	Description
wallet_instance_attestation	It MUST be set to a value containing the *Wallet Instance Attestation* JWT.
credentials_backup	Object that describes specifics of the:term:*Credential* that are backed up. This object contains a list of name/value pairs, where each name is a unique identifier of the *Credential Issuer*. This identifier is used to initiate the issuance phase. The value is an array of unique strings. Each string corresponds to the `credential_configuration_id` that identifies the specific:term:*Digital Credential* that was issued.

Restore flow for Hardware Binding Credential

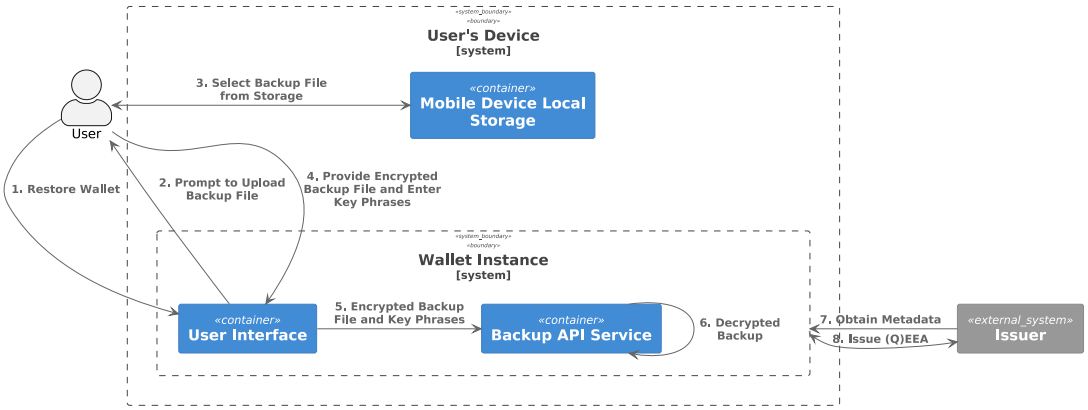


Fig. 10.9: Restore Flow

Considering that the *User* has initialized the new *Wallet Instance* and it is in active state by obtaining a new PID, this specification relaxes the requirement of the ARF concerning the addition of the PID in the backup file.

In the same way as the PID, IT-Wallet ID also MUST not be included in the backup file.

Below, the description of the steps of Fig. 10.9:

Steps 1-6: The *User* wants to restore the Digital Credentials using the backup previously created with their *Wallet Instance*. The *User* selects *restore Digital Credentials backup* in the *Wallet Instance* app and a prompt with the import function is provided to the User (*WP_126* (page 765)). The backup file to be imported can be provided using a local storage or a remote location using a cloud storage as well, and therefore submit the recovery key phrases (*WP_127* (page 765)). To check the authenticity of the file, the *Wallet Instance* MUST verify the backup JWT's signature to ensure its authenticity (*WP_129* (page 765)). To do this, it first extracts the *Wallet Instance Attestation* JWT from `wallet_instance_attestation` claim and obtains the related public key using the *Wallet Instance Attestation* (`cnf` claim) as per *WP_128* (page 765).

Steps 7-8: The *Wallet Instance* for each HW binding credentials entry in the payload of the backup JWT performs the following steps (*WP_130* (page 765)):

- It extracts the *Credential Issuer* identifier and the `credential_configuration_id` from the entry. The former is used to identify the Issuer and obtains its metadata, while the latter will be used to signal the Credential type to the *Credential Issuer* (*WP_130a* (page 765)).
- Using the *Issuer* identifier the *Wallet Instance* obtains the metadata of the *Credential Issuer* and makes an issuance request to the *Credential Issuer* by providing the new *Holder Key Binding* with the *Credential* (*WP_130b* (page 765)).

Note

During restore, the *Wallet Instance* MUST perform a Wallet-Initiated Authorization Code Issuance Flow as defined in Section *Low-Level Issuance Flow* (page 422), using a fresh *Holder Key Binding*. For (Q)EAA, this includes the presentation gate at the *Credential Issuer* acting as a Relying Party, according to credential-issuance-endpoint:User Authentication Method Selection. This MUST NOT be interpreted as the Re-Issuance Flow defined in Section *Re-Issuance Flow* (page 448), which applies only to same-instance Credential refresh (`UPDATE` / `ATTRIBUTE_UPDATE`) and requires a Refresh Token bound to the original *Wallet Instance*. From the *Credential Issuer*'s perspective, restore is indistinguishable from a first-time Wallet-Initiated issuance. Credentials restored from distinct Issuers REQUIRE a separate Authorization Code Issuance Flow per Issuer.

Note

The *Wallet Instance* MUST NOT check the expiration of the *Wallet Instance Attestation* as its main purpose is to enable the *Wallet Instance* to verify the authenticity of the backup file by ensuring it has been created and signed by a *Wallet Instance* of a specific *Wallet Provider* (*WP_128a* (page 765)).

10.1.6 Wallet Provider Entity Configuration

An HTTP GET request to the Federation endpoint allows the retrieval of the Wallet Provider Entity Configuration (*WP_001* (page 737)).

The returned Entity Configuration of the Wallet Provider MUST contain the attributes described in the sections below.

The Wallet Provider Entity Configuration is a signed JWT containing the public keys and supported algorithms of the Wallet Solution as a component of the Wallet Provider. It is

structured in accordance with the OID-FED and *The Infrastructure of Trust* (page 109) outlined in this specification (*WP_002* (page 737)).

Wallet Provider Entity Configuration JWT Header

Key	Value
alg	Algorithm used to verify the token signature. It MUST be one of the possible values indicated in *Cryptographic Algorithms* (page 615) (e.g., ES256).
kid	Thumbprint of the public key used for the signature.
typ	Media type, set to `entity-statement+jwt`.

Wallet Provider Entity Configuration JWT Payload

Key	Value
`iss`	REQUIRED. Public URL of the Wallet Solution.
`sub`	REQUIRED. Public URL of the Wallet Solution.
`iat`	REQUIRED. Issuance datetime in Unix Timestamp format.
`exp`	REQUIRED. Expiration datetime in Unix Timestamp format.
`authority_hints`	REQUIRED. Array of URLs (String) containing the list of URLs of the immediate superior Entities, such as the Trust Anchor or an Intermediate, that MAY issue an Entity Statement related to the Wallet Solution.
`jwks`	REQUIRED. A JSON Web Key Set (JWKS) representing the public part of the Federation Entity signing keys. The corresponding private key is used by the Wallet Solution to sign the Entity Configuration about itself.
`metadata`	REQUIRED.JSON object that represents the Entity's Types and the metadata for those Entity Types. Each member name of the JSON object is an Entity Type Identifier, and each value MUST be a JSON object containing metadata parameters according to the metadata schema of the Entity Type. It MUST contain the `openid_wallet_provider` and the `federation_entity` metadata.

Note

Tests covering the Entity Configuration structure (header and payload) are provided in *WP_002a–002h* (page 737).

Example of a Wallet Provider Entity Configuration

Below is a non-normative example of a payload of a Wallet Provider Entity Configuration containing a metadata for

- *federation_entity*
- *openid_wallet_provider*


```
{
  "iss": "https://wallet-solution.example.org",
  "sub": "https://wallet-solution.example.org",
  "iat": 1687171759,
  "exp": 1709290159,
  "jwks": {
    "keys": [
      {
        "crv": "P-256",
        "kty": "EC",
        "x": "qrJrj3Af_B57sbOIRrcBM7br7wOc8ynj7lHFPTeffUk",
        "y": "1H0cWDyGgvU8w-kPKU_xycOCUNT2o0bwslIQtnPU6iM",
        "kid": "5t5YYpBhN-EgIEEI5iUzr6r0MR02LnVQ0OmekmNKcjY"
      }
    ]
  },
  "metadata": {
    "openid_wallet_provider": {
      "logo_uri": "https://wallet-solution.example.org/compact-logo.svg",
      "jwks": {
        "keys": [
          {
            "crv": "P-256",
            "kty": "EC",
            "x": "BxYsu3QvYmOz1fl1l5hGyPWlpvgTzz3AY3j3K_9zGPs",
            "y": "ob34Wmfah_ScQXaYMJWoBkZSwO-kQ0VTgMk4VZfu48w",
            "kid": "749b495837819c00cfee1749b495837819c00cfee1"
          }
        ]

      },
      "wallet_metadata": {
        "wallet_name": "Wallet X",
        "authorization_endpoint": "https://wallet-solution.digital-strategy.europa.eu/authorization",
        "credential_offer_endpoint": "https://wallet-solution.digital-strategy.europa.eu/credential_offer",
        "response_types_supported": [
          "vp_token"
```

```
        ],
        "vp_formats_supported": {
          "dc+sd-jwt": {
            "sd-jwt_alg_values": [
              "ES256",
              "ES384"
            ]
          }
        },
        "request_object_signing_alg_values_supported": [
          "ES256"
        ],
        "client_id_prefixes_supported": ["openid_federation", "x509_hash"]
      }
    },
    "federation_entity": {
      "organization_name": "IT-Wallet Provider",
      "homepage_uri": "https://wallet-provider.example.org",
      "policy_uri": "https://wallet-provider.example.org/privacy_policy",
      "tos_uri": "https://wallet-provider.example.org/info_policy",
      "logo_uri": "https://wallet-provider.example.org/logo.svg"
    }
  },
  "authority_hints": [
    "https://registry.eudi-wallet.example.it"
  ]
}
```


10.1.7 Wallet Solution Metadata

The metadata JSON Object whose key is `openid_wallet_provider` contains the following parameters. The public keys found in this object are exclusively used for signing and/or encryption operations required by this Entity when acting as a component of the Wallet Provider (e.g., sign the Wallet Attestations to the Wallet Instance).

Key	Value
`jwks`	CONDITIONAL. JSON Web Key Set document, passed by value, containing the Entity's keys for that Entity Type. It MUST be present if `jwks_uri` and `signed_jwks_uri` are absent.
`jwks_uri`	CONDITIONAL. URL referencing a JWK Set document containing the Wallet Solution's keys for that Entity Type. This URL MUST use the https scheme. It MUST be present if `jwks` and `signed_jwks_uri` are absent.

continues on next page

Table 10.5 – continued from previous page

Key	Value
`signed_jwks_uri`	CONDITIONAL. URL referencing a signed JWT having the Entity's JWK Set document for that Entity Type as its payload. This URL MUST use the https scheme. The JWT MUST be signed using a Federation Entity Key. A successful response from the URL MUST use the HTTP status code 200 with the Content Type `application/jwk-set+jwt`. It MUST be present if `jwks` and `jwks_uri` are absent.
`logo_uri`	REQUIRED. URL of the entity's logo that will be shown to the User. The logo mime type MUST be `application/svg`.
`wallet_metadata`	REQUIRED. It contains the Wallet Metadata parameters as defined in *Table of Wallet Metadata Parameters* (page 496) and the following two parameters • `credential_offer_endpoint` Credential Offer Endpoint of a Wallet. • `wallet_name` String containing a human-readable name of the Wallet.

Note

There are Digital Credential flows that require retrieving Wallet Solution metadata before interacting with the Wallet itself. The flow is described in *Wallet Metadata Retrieval Flow* (page 541).

10.2 Credential Issuer Solution

An Issuer, as an Organizational Entity participating in the IT-Wallet ecosystem, MUST provide Technical Solutions (Issuer Solution) that combine software, hardware, services, settings, and configurations to issue Digital Credentials to User Wallet Instances in a secure and trusted manner.

The following diagram depicts the Issuer Solution High Level Architecture.

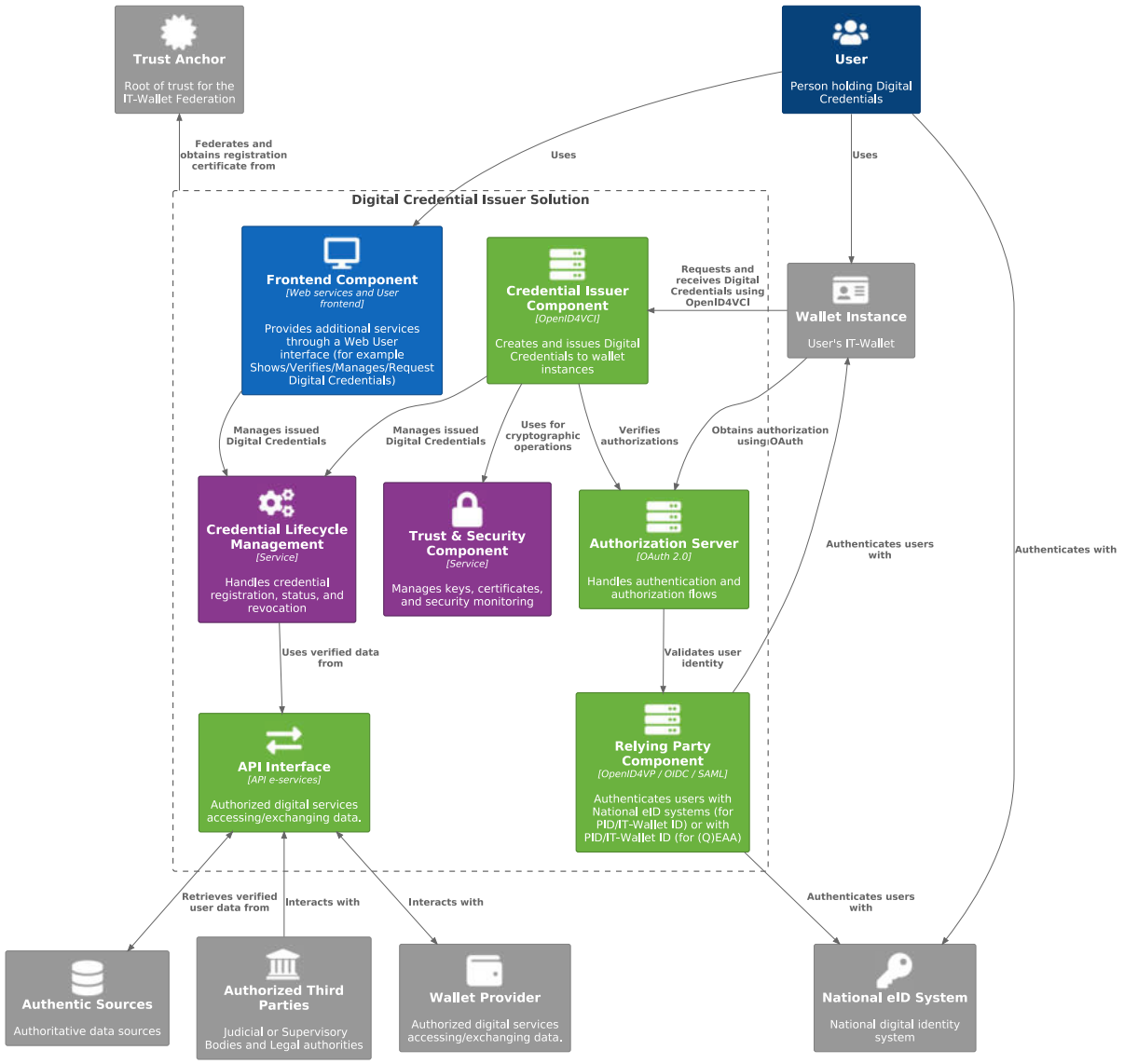


Fig. 10.10: Credential Issuer Solution High Level Architecture.

10.2.1 Credential Issuer Requirements

The Digital Credential Issuer Solution MUST:

1. Register with the Federation Authority to obtain proper authorization for issuing specific credential types.
2. Implement secure creation and issuance mechanisms that ensure integrity and confidentiality.
3. Communicate with Authentic Sources through secure and reliable API Services to obtain verified User data.
4. Authenticate to Wallet Instances during issuance to prove its legitimacy.
5. Support immediate issuance flow and MAY support deferred issuance for various operational scenarios.
6. Implement appropriate error handling and User notifications for all processes.
7. Maintain comprehensive audit trails while respecting privacy regulations.
8. Issue Digital Credentials that support Selective Disclosure.
9. Periodically renew its trust with the Federation.
10. Register the Relying Party Component within the CIEid Digital Identity Federation ecosystem (for PID and IT-Wallet ID issuance), and within the IT-Wallet ecosystem (for (Q)EAA issuance, if required).
11. Authenticate Users for PID, IT-Wallet ID and (Q)EAA issuance according to credential-issuance-endpoint:User Authentication Method Selection.
12. Implement proper procedures for the entire Digital Credential lifecycle as detailed in Section *Digital Credential Lifecycle* (page 398).

For the Frontend Component (if implemented):

13. Authenticate Users with a Level of Assurance (LoA) at least equal to that used to obtain the Digital Credential being issued or managed.
14. Provide appropriate security measures to protect User data and Digital Credential information.

10.2.2 Component Details

Frontend Component

The Frontend Component, if provided by the Issuer, MUST provide a web-based User interface for Digital Credential management, offering functionality to:

- Display and verify issued Digital Credentials and their status.
- Manage Digital Credential lifecycle (e.g., revocation).
- Initiate issuance through Credential Offers.
- Provide User support and documentation.

Issuers MAY provide additional services to the User through the Frontend Component. These additional services MUST NOT conflict with any regulatory or technical requirements defined in this technical specification or in national/European security and privacy regulations.

Credential Issuer Component

Following the OpenID4VCI specification and the implementation profile in Section *Digital Credential Issuance* (page 419), this component MUST:

- Issue Digital Credentials to Wallet Instances.
- Process Digital Credential requests.
- Obtain User data from Authentic Sources.
- Generate properly formatted and signed Digital Credentials in supported formats (SD-JWT-VC, mDoc-CBOR). See Section *Digital Credential Data Model* (page 357) for more details.
- Implement the Digital Credential issuance protocols and flows.

Authorization Server

This OAuth 2.0 based component MUST:

- Handle authentication and authorization flows.
- Manage access/refresh tokens and authorization codes.
- Validate User identity confirmed by the Relying Party Component.

Relying Party Component

When User authentication is required, this component MUST authenticate Users according to credential-issuance-endpoint:User Authentication Method Selection.

API Interface

This component MUST establish secure connections with Authentic Sources to:

- Retrieve verified User data.
- Properly authenticate and authorize connections.
- Format data according to Digital Credential schemas.
- Provide cryptographic evidence of User authentication when required.

Note

For public Authentic Sources, a Credential Issuer MUST use PDND according to rules in Sections *e-Service PDND* (page 675), *Status Update by Authentic Sources* (page 408), and *e-Service PDND Authentic Source Catalog* (page 602).

Credential Lifecycle Management

This component MUST handle:

- Status management (maintaining and updating validity).
- Revocation processes (implementing mechanisms to revoke or suspend Digital Credentials), according to Section *Digital Credential Lifecycle* (page 398).
- Renewal workflows (managing Digital Credential renewal processes), according to the mechanisms defined in Section *Digital Credential Issuance* (page 419).

Trust & Security Component

This component MUST ensure security through:

- Key and certificate management.
- Audit logging.
- Security monitoring and incident response.
- Compliance with IT-Wallet Federation security requirements.

10.2.3 Interaction Patterns

The Digital Credential Issuer Solution supports these interaction patterns:

1. **User to Frontend**: Web-based interactions for Digital Credential management.
2. **Frontend to Credential Issuer**: Converts user requests into OpenID4VCI protocol messages.
3. **Wallet Instance to Credential Issuer**: Direct protocol-based interactions following the issuance flow.
4. **Relying Party to Identity Providers**: Authentication interactions with national eID systems or PID/IT-Wallet ID verification.
5. **API Interface to Authentic Sources**: Secure API calls to retrieve verified User data.

All interactions must follow the security considerations in Section *Digital Credential Issuance* (page 419), including proper handling of tokens, proofs, and cryptographic materials.

10.2.4 Credential Issuer Entity Configuration

The Credential Issuers, as Federation Entity, MUST adhere to the guidelines outlined in Section *Configuration of the Federation* (page 116). Specifically, they MUST provide a *well-known* endpoint that hosts their Entity Configuration. The Entity Configuration of Credential Issuers MUST contain the parameters defined in the Sections *Entity Configuration Leaves and Intermediates* (page 120) and *Entity Configurations Common Parameters* (page 118).

The Credential Issuers MUST provide, at least, the following metadata types:

- *federation_entity*
- *oauth_authorization_server*
- *openid_credential_issuer*

In cases where the (Q)EAA Providers authenticate Users using their Wallet Instance, then the metadata for *openid_credential_verifier* MUST be provided in addition to the metadata above. In case a national eID scheme is used by the Credential Issuers for the User authentication, they MAY include a metadata for *openid_relying_party* within their Entity Configuration. The *openid_relying_party* metadata MUST be compliant with the Technical Specification SPID/CIE-OpenID-Connect-Specifications.

The *federation_entity* metadata MUST contain the parameters as defined in Section *Metadata of federation_entity Leaves* (page 122).

The *oauth_authorization_server* metadata MUST contain the parameters as defined in Section credential-issuer-metadata:Metadata for oauth_authorization_server.

The *openid_credential_issuer* metadata MUST contain the parameters as defined in Section credential-issuer-metadata:Metadata for openid_credential_issuer.

The *openid_credential_verifier* metadata MUST contain the parameters as defined in Section *Relying Party Entity Configuration* (page 352).

Example of a (Q)EAA Provider Entity Configuration

Below is a non-normative example of an Entity Configuration of a (Q)EAA Provider containing a metadata for

- *federation_entity*
- *oauth_authorization_server*
- *openid_credential_issuer*
- *openid_credential_verifier*


```
{
    "iat": 1718207217,
    "exp": 1749743216,
    "iss": "https://eaa-provider.example.org",
    "sub": "https://eaa-provider.example.org",
    "authority_hints": [
        "https://trust-anchor.example.org"
    ],
    "jwks": {
        "keys": [
            {
                "kid": "FANFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
            }
        ]
    },
    "metadata": {
        "federation_entity": {
            "homepage_uri": "https://eaa-provider.example.org/",
            "organization_name": "Organization Name",
            "contacts": [
                "informazioni@example.it",
                "protocollo@pec.example.it"
            ],
            "tos_uri": "https://eaa-provider.example.org/public/info_policy.html",
            "policy_uri": "https://eaa-provider.example.org/public/privacy_policy.html",
```

```
        "logo_uri": "https://eaa-provider.example.org/public/logo.svg"
    },
    "oauth_authorization_server": {
        "issuer": "https://eaa-provider.example.org",
        "pushed_authorization_request_endpoint": "https://eaa-provider.example.org/as/par",
        "authorization_endpoint": "https://eaa-provider.example.org/authorize",
        "token_endpoint": "https://eaa-provider.example.org/token",
        "client_registration_types_supported": [
            "automatic"
        ],
        "code_challenge_methods_supported": [
            "S256"
        ],
        "acr_values_supported": [
            "https://trust-anchor.example.it/loa/substantial",
            "https://trust-anchor.example.it/loa/high"
        ],
        "scopes_supported": [
            "EuropeanDisabilityCard",
            "mDL"
        ],
        "response_types_supported": [
            "code"
        ],
        "authorization_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "grant_types_supported": [
            "authorization_code"
        ],
        "token_endpoint_auth_methods_supported": [
            "attest_jwt_client_auth"
        ],
        "client_attestation_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "client_attestation_pop_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "token_endpoint_auth_signing_alg_values_supported": [
            "ES256",
```

```
            "ES384",
            "ES512"
        ],
        "request_object_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "dpop_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "jwks": {
            "keys": [
                {
                    "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                    "kty": "EC",
                    "crv": "P-256",
                    "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                    "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                }
            ]
        }
    },
    "openid_credential_issuer": {
        "credential_issuer": "https://eaa-provider.example.org",
        "credential_endpoint": "https://eaa-provider.example.org/credential",
        "nonce_endpoint": "https://eaa-provider.example.org/nonce-endpoint",
        "deferred_credential_endpoint": "https://eaa-provider.example.org/deferred-credenti
al",
        "notification_endpoint": "https://eaa-provider.example.org/notification",
        "display": [
            {
                "name": "EAA Provider",
                "locale": "it-IT",
                "logo": {
                    "uri": "https://eaa-provider.example.org/public/compact-logo-IT.svg",
                    "uri#integrity": "2Rltygo7bKBE4E498tb/lLn9DIkimBhrzrsUERyUQjo=",
                    "alt_text": "Testo alternativo all'immagine del logo"
                }

            },
            {
                "name": "EAA Provider",
                "locale": "en-US",
                "logo": {
```

```
                    "uri": "https://eaa-provider.example.org/public/compact-logo-US.svg",
                    "uri#integrity": "mBQEi42aj0VYnFngpOjbqJGkuCnB5LmbZRnWpO+AHqUBVQxOe9kMI
POz+pSjRArE",
                    "alt_text": "Alternative text for the logo image"
                }
            }
        ],
        "credential_configurations_supported": {
            "dc_sd_jwt_EuropeanDisabilityCard": {
                "format": "dc+sd-jwt",
                "scope": "EuropeanDisabilityCard",
                "cryptographic_binding_methods_supported": [
                    "jwk"
                ],
                "credential_signing_alg_values_supported": [
                    "ES256",
                    "ES384",
                    "ES512"
                ],
                "proof_types_supported": {
                    "jwt": {
                        "proof_signing_alg_values_supported": [
                            "ES256",
                            "ES384",
                            "ES512"
                        ],
                        "key_attestations_required": {
                        "key_storage": [ "iso_18045_enhanced-basic" ],
                        "user_authentication": [ "iso_18045_enhanced-basic" ]
                        }
                    }
                },
                "vct": "urn:eudi:EuropeanDisabilityCard:it:1",
                "schema_id": "EuropeanDisabilityCard+dc+sd-jwt+urn:eudi:EuropeanDisabilityC
ard:it:1",
                "authentic_sources": [
                    {
                        "entity_id": "https://authentic-source.example.com",
                        "dataset_id": "12345"
                    }

                ],
                "credential_metadata": {
                    "display": [
                        {
                            "name": "Carta della disabilità europea",
                            "locale": "it-IT",
```

```
                "description": "Versione digitale della carta della disabilità
europea",
                "logo": {
                    "uri": "https://trust-registry.it-wallet.example.it/logos/E
uropeanDisabilityCard/logo-IT.svg",
                    "uri#integrity": "3GGZJH7igkyeprXI9Plm3dQsWMYOeuz0MaEK34PC7
sOtLfgeoK9scXOvMeiACF8Tk2WOT36NsYCPduoLyOD1Sg==",
                    "alt_text": "Testo alternativo all'immagine del logo"
                },
                "background_color": "#12107c"
            },
            {
                "name": "European Disability Card",
                "locale": "en-US",
                "description": "Digital version of the European Disability Card
",
                "logo": {
                    "uri": "https://trust-registry.it-wallet.example.it/logos/E
uropeanDisabilityCard/logo-US.svg",
                    "uri#integrity": "3GGZJH7igkyeprXI9Plm3dQsWMYOeuz0MaEK34PC7
sOtLfgeoK9scXOvMeiACF8Tk2WOT36NsYCPduoLyOD1Sg==",
                    "alt_text": "Alternative text for the logo image"
                },
                "background_color": "#12107c"
            }
        ],
        "claims": [
            {
                "path": ["document_number"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Numero Documento",
                        "locale": "it-IT",
                        "description": "Numero Documento della carta europea de
lla disabilità"
                    },
                    {
                        "name": "Document Number",
                        "locale": "en-US",
                        "description": "Document number of the European Disabil
ity Card"
                    }
                ]
            },
            {
```

```
            "path": ["given_name"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Nome",
                    "locale": "it-IT",
                    "description": "Nome del titolare della carta"
                },
                {
                    "name": "Name",
                    "locale": "en-US",
                    "description": "Cardholder name"
                }
            ]
        },
        {
            "path": ["family_name"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Cognome",
                    "locale": "it-IT",
                    "description": "Cognome del titolare della carta"
                },
                {
                    "name": "Family Name",
                    "locale": "en-US",
                    "description": "cardholder last name"
                }
            ]
        },
        {
            "path": ["birth_date"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Data di Nascita (YYYY-MM-GG)",
                    "locale": "it-IT",
                    "description": "Data di nascita del titolare della cart
a"

                },
                {
                    "name": "Date of Birth (YYYY-MM-GG)",
```

```
                        "locale": "en-US",
                        "description": "cardholder birth date"
                    }
                ]
            },
            {
                "path": ["personal_administrative_number"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Codice Fiscale",
                        "locale": "it-IT",
                        "description": "Codice fiscale del titolare della carta
"
                    },
                    {
                        "name": "Tax Identification Number",
                        "locale": "en-US",
                        "description": "cardholder tax identification number"
                    }
                ]
            },
            {
                "path": ["expiry_date"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Data di Scadenza (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di scadenza della carta"
                    },
                    {
                        "name": "Expiration Date (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Card expiration date"
                    }
                ]
            },
            {
                "path": ["constant_attendance_allowance"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Diritto accompagnatore",
```

```
                                    "locale": "it-IT",
                                    "description": "Questo campo indica se il titolare dell
a carta ha diritto all'indennità di accompagnamento"
                                },
                                {
                                    "name": "Constant attendance allowance",
                                    "locale": "en-US",
                                    "description": "This field indicates whether the cardho
lder is entitled to the accompanying allowance"
                                }
                            ]
                        },
                        {
                            "path": ["portrait"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Foto",
                                    "locale": "it-IT",
                                    "description": "Fotografia del titolare della carta"

                                },
                                {
                                    "name": "Portrait",
                                    "locale": "en-US",
                                    "description": "Photo of the cardholder"
                                }
                            ]
                        },
                        {
                            "path": ["link_qr_code"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Link QR Code",
                                    "locale": "it-IT",
                                    "description": "Questo campo contiene il QR-Code"
                                },
                                {
                                    "name": "Link QR Code",
                                    "locale": "en-US",
                                    "description": "This field contains the QR-Code"
                                }
                            ]
                        }
```

```
                    ]
                }
            },
            "dc_sd_jwt_mDL": {
                "format": "dc+sd-jwt",
                "scope": "mDL",
                "cryptographic_binding_methods_supported": [
                    "jwk"
                ],
                "credential_signing_alg_values_supported": [
                    "ES256",
                    "ES384",
                    "ES512"
                ],
                "proof_types_supported": {
                    "jwt": {
                        "proof_signing_alg_values_supported": [
                            "ES256",
                            "ES384",
                            "ES512"
                        ]
                    }
                },
                "vct": "urn:eudi:mDL:it:1",
                "schema_id": "mDL+dc+sd-jwt+urn:eudi:mDL:it:1",
                "authentic_sources": [
                    {
                        "entity_id": "https://authentic-source.example.com",
                        "dataset_id": "12345"
                    }

                ],
                "credential_metadata": {
                    "display": [
                        {
                            "name": "Patente di guida",
                            "locale": "it-IT",
                            "description": "Versione digitale della patente di guida",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-IT.svg",
                                "uri#integrity": "qU4OnSrnRcL695rQqy9ayCMx6sboaXxEaW5rXvSAm
nQ=",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        },
```

```
                        {
                            "name": "Mobile Driver's License",
                            "locale": "en-US",
                            "description": "Digital version of Mobile Driver's License",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-US.svg",
                                "uri#integrity": "F5P1wXpfToammXkx6HF++66G534ystjOy7r4UMWwz
1lcZYJU8kAKRKtFua7DvZLe",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        }
                    ],
                    "claims": [
                        {
                            "path": ["given_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Nome",
                                    "locale": "it-IT",
                                    "description": "Nome del titolare della patente"
                                },
                                {
                                    "name": "First Name",
                                    "locale": "en-US",
                                    "description": "Name of the driving license holder"
                                }
                            ]
                        },
                        {
                            "path": ["family_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Cognome",
                                    "locale": "it-IT",
                                    "description": "Cognome del titolare della patente"
                                },
                                {
                                    "name": "Family Name",
                                    "locale": "en-US",
                                    "description": "last name of the driving license holder
"
```

```
                        }
                    ]
                },
                {
                    "path": ["birth_date"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Data di nascita (YYYY-MM-GG)",
                            "locale": "it-IT",
                            "description": "Data di nascita del titolare della pate
nte"
                        },
                        {
                            "name": "Date of Birth (YYYY-MM-GG)",
                            "locale": "en-US",
                            "description": "Date of birth of the driving license ho
lder"
                        }
                    ]
                },
                {
                    "path": ["place_of_birth"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Luogo di Nascita",
                            "locale": "it-IT",
                            "description": "Luogo di nascita del titolare della pat
ente"
                        },
                        {
                            "name": "Place of Birth",
                            "locale": "en-US",
                            "description": "Place of birth of the driving license h
older"
                        }
                    ]
                },
                {
                    "path": ["issue_date"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
```

```
                        "name": "Data di rilascio (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di rilascio della patente"
                    },
                    {
                        "name": "Issue Date (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Date of issuance of the driving license
"
                    }
                ]
            },
            {
                "path": ["expiry_date"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Data di scadenza (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di scadenza della patente"
                    },
                    {
                        "name": "Expiry Date (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Date of expiry of the driving license"
                    }
                ]
            },
            {
                "path": ["issuing_country"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Paese di rilascio",
                        "locale": "it-IT",
                        "description": "Paese di rilascio della patente"

                    },
                    {
                        "name": "Issuing Country",
                        "locale": "en-US",
                        "description": "Issuing country of the driving license"
                    }
                ]
            },
```

```
                {
                    "path": ["issuing_authority"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Autorità di rilascio",
                            "locale": "it-IT",
                            "description": "Autorità che ha rilasciato la patente"
                        },
                        {
                            "name": "Issuing Authority",
                            "locale": "en-US",
                            "description": "Issuing authority of the driving licens
e"
                        }
                    ]
                },
                {
                    "path": ["document_number"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Numero di documento",
                            "locale": "it-IT",
                            "description": "Numero Documento della patente"
                        },
                        {
                            "name": "Document Number",
                            "locale": "en-US",
                            "description": "Document number of the driving license"
                        }
                    ]
                },
                {
                    "path": ["portrait"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Foto codificata",
                            "locale": "it-IT",
                            "description": "Fotografia del titolare della patente"
                        },
                        {
                            "name": "Portrait base64 encoded",
```

```
                                "locale": "en-US",
                                "description": "Photo of of the driving licence holder"
                            }
                        ]
                    },
                    {
                        "path": ["driving_privileges"],
                        "mandatory": "true",
                        "sd": "always",
                        "display": [
                            {
                                "name": "Abilitazioni alla guida",
                                "locale": "it-IT",
                                "description": "Elenco delle categorie di abilitazione"
                            },
                            {
                                "name": "Driving Privileges",
                                "locale": "en-US",
                                "description": "Driving Privileges list"
                            }
                        ]
                    },
                    {
                        "path": ["restrictions_conditions"],
                        "mandatory": "true",
                        "sd": "always",
                        "display": [
                            {
                                "name": "Annotazioni/Restrizioni",
                                "locale": "it-IT",
                                "description": "Annotazioni/Restrizioni valide per tutt
e le categorie separate da spazio"
                            },
                            {
                                "name": "Restrictions/Conditions",
                                "locale": "en-US",
                                "description": "Restriction/Condition for all driving p
rivileges separated by space"
                            }
                        ]
                    },
                    {
                        "path": ["driving_privileges_details"],
                        "mandatory": "true",
                        "sd": "always",
                        "display": [
                            {
```

```
                                    "name": "Dettagli delle categorie di abilitazione",
                                    "locale": "it-IT",
                                    "description": "Dettagli relativi alle specifiche categ
orie di abilitazione"
                                },
                                {
                                    "name": "Driving privilege details",
                                    "locale": "en-US",
                                    "description": "Details related to the specific driving
privileges"
                                }
                            ]
                        }
                    ]
                }
            },
            "mso_mdoc_mDL": {
                "format": "mso_mdoc",
                "scope": "mDL",
                "doctype": "org.iso.18013.5.1.mDL",
                "schema_id": "mDL+mso_mdoc+org.iso.18013.5.1.mDL",
                "authentic_sources": [
                    {
                        "entity_id": "https://authentic-source.example.com",
                        "dataset_id": "12345"
                    }

                ],
                "cryptographic_binding_methods_supported": [
                    "cose_key"
                ],
                "credential_signing_alg_values_supported": [
                    -7,
                    -35,
                    -36
                ],
                "credential_metadata": {
                    "display": [
                        {
                            "name": "Patente di guida",
                            "locale": "it-IT",
                            "description": "Versione digitale della patente di guida",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-IT.svg",
                                "uri#integrity": "qU4OnSrnRcL695rQqy9ayCMx6sboaXxEaW5rXvSAm
nQ=",
```

```
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        },
                        {
                            "name": "Mobile Driver's License",
                            "locale": "en-US",
                            "description": "Digital version of Mobile Driver's License",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-US.svg",
                                "uri#integrity": "F5P1wXpfToammXkx6HF++66G534ystjOy7r4UMWwz
1lcZYJU8kAKRKtFua7DvZLe",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        }
                    ],
                    "claims": [
                        {
                            "path": ["org.iso.18013.5.1", "given_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Nome",
                                    "locale": "it-IT",
                                    "description": "Nome del titolare della patente"
                                },
                                {
                                    "name": "First Name",
                                    "locale": "en-US",
                                    "description": "Name of the driving license holder"
                                }
                            ]
                        },
                        {
                            "path": ["org.iso.18013.5.1", "family_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Cognome",
                                    "locale": "it-IT",
                                    "description": "Cognome del titolare della patente"
                                },
                                {
```

```
                                "name": "Family Name",
                                "locale": "en-US",
                                "description": "last name of the driving license holder
"
                            }
                        ]
                    },
                    {
                        "path": ["org.iso.18013.5.1", "birth_date"],
                        "mandatory": "true",
                        "sd": "always",
                        "display": [
                            {
                                "name": "Data di nascita (YYYY-MM-GG)",
                                "locale": "it-IT",
                                "description": "Data di nascita del titolare della pate
nte"
                            },
                            {
                                "name": "Date of Birth (YYYY-MM-GG)",
                                "locale": "en-US",
                                "description": "Date of birth of the driving license ho
lder"
                            }
                        ]
                    },
                    {
                        "path": ["org.iso.18013.5.1", "birth_place"],
                        "mandatory": "true",
                        "sd": "always",
                        "display": [
                            {
                                "name": "Luogo di Nascita",
                                "locale": "it-IT",
                                "description": "Luogo di nascita del titolare della pat
ente"
                            },
                            {
                                "name": "Place of Birth",
                                "locale": "en-US",
                                "description": "Place of birth of the driving license h
older"
                            }
                        ]
                    },
                    {
                        "path": ["org.iso.18013.5.1", "issue_date"],
```

```
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Data di rilascio (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di emissione della patente"
                    },
                    {
                        "name": "Issue Date (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Date of issuance of the driving license
"
                    }
                ]
            },
            {
                "path": ["org.iso.18013.5.1", "expiry_date"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Data di scadenza (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di scadenza della patente"
                    },
                    {
                        "name": "Expiry Date (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Date of expiry of the driving license"
                    }
                ]
            },
            {
                "path": ["org.iso.18013.5.1", "issuing_country"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Paese di rilascio",
                        "locale": "it-IT",
                        "description": "Paese di rilascio della patente"
                    },
                    {
                        "name": "Issuing Country",
                        "locale": "en-US",
                        "description": "Issuing country of the driving license"
```

```
                    }
                ]
            },
            {
                "path": ["org.iso.18013.5.1", "issuing_authority"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Autorità di rilascio",
                        "locale": "it-IT",
                        "description": "Autorità che ha rilasciato la patente"
                    },
                    {
                        "name": "Issuing Authority",
                        "locale": "en-US",
                        "description": "Issuing authority of the driving licens
e"
                    }
                ]
            },
            {
                "path": ["org.iso.18013.5.1", "document_number"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Numero di documento",
                        "locale": "it-IT",
                        "description": "Numero Documento della patente"
                    },
                    {
                        "name": "Document Number",
                        "locale": "en-US",
                        "description": "Document number of the driving license"
                    }
                ]
            },
            {
                "path": ["org.iso.18013.5.1", "portrait"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Foto codificata",
                        "locale": "it-IT",
                        "description": "Fotografia del titolare della patente"
```

```
                                },
                                {
                                    "name": "Portrait",
                                    "locale": "en-US",
                                    "description": "Photo of of the driving licence holder"
                                }
                            ]
                        },
                        {
                            "path": ["org.iso.18013.5.1", "driving_privileges"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Abilitazioni alla guida",
                                    "locale": "it-IT",
                                    "description": "Elenco delle categorie di abilitazione"
                                },
                                {
                                    "name": "Driving Privileges",
                                    "locale": "en-US",
                                    "description": "Driving Privileges list"
                                }
                            ]
                        },
                        {
                            "path": ["org.iso.18013.5.1", "un_distinguishing_sign"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Codice identificativo della Nazione",
                                    "locale": "it-IT",
                                    "description": "Segno distintivo del paese che emette l
a patente"
                                },
                                {
                                    "name": "Distinguishing sign of the issuing country",
                                    "locale": "en-US",
                                    "description": "Distinguishing sign of the issuing coun
try"
                                }
                            ]
                        }
                    ]
                }
            }
```

```
            },
            "jwks": {
                "keys": [
                    {
                        "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                        "kty": "EC",
                        "crv": "P-256",
                        "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                        "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                    }
                ]
            },
            "trust_frameworks_supported": [
                "it_cie",
                "it_wallet",
                "eudi_wallet"
            ],
            "status_list_aggregation_endpoint": "https://eaa-provider.example.org/statuslists/aggregation"
        },
        "openid_credential_verifier": {
            // see relying party metadata section and endpoints
        },
        "jwks": {
            "keys": [
                {
                    "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                    "kty": "EC",
                    "crv": "P-256",
                    "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                    "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                }
            ]
        }
    }
}
}
```

10.2.5 Credential Issuer Metadata

Metadata for oauth_authorization_server

The *oauth_authorization_server* metadata MUST contain the following parameters.

Claim	Description
issuer	It MUST contain an HTTPS URL that uniquely identifies the Credential Issuer.
pushed_authorization_request_endpoint	The URL of the pushed authorization request endpoint is where a Wallet Instance MUST submit an authorization request to obtain a *request_uri* value, which can then be used at the authorization endpoint. See **RFC 9126#as_metadata**.
authorization_endpoint	URL of the authorization server's authorization endpoint. See **RFC 8414#section-2**.
token_endpoint	URL of the authorization server's token endpoint. See **RFC 8414#section-2**.
client_registration_types_supported	Array specifying the registration types supported. The authorization server MUST support *automatic*. See OID-FED Section 5.1.3.
code_challenge_methods_supported	JSON array containing a list of Proof Key for Code Exchange (PKCE) **RFC 7636** code challenge methods supported by the authorization server. The authorization server MUST support *S256*.
acr_values_supported	See OpenID Connect Discovery 1.0 Section 3. The supported values are: • *https://{Trust Anchor Domain}/loa/low* • *https://{Trust Anchor Domain}/loa/substantial* • *https://{Trust Anchor Domain}/loa/high*
scopes_supported	JSON array containing a list of the supported *scope* values. See **RFC 8414#section-2**.
response_types_supported	JSON array containing a list of the supported "response_type" values, as specified in **RFC 8414**. The supported value MUST be *code*.
authorization_signing_alg_values_supported	JSON array containing a list of the **RFC 7515** supported signing algorithms (*alg* values). The values MUST be set according to Section *Cryptographic Algorithms* (page 615).
grant_types_supported	JSON array containing a list of the supported grant type values. The authorization server MUST support *authorization_code*.
token_endpoint_auth_methods_supported	JSON array containing a list of supported client authentication methods. The Token Endpoint MUST support *attest_jwt_client_auth* as defined in OAUTH-ATTESTATION-CLIENT-AUTH.
client_attestation_signing_alg_values_supported	JSON array containing the list of JWS "alg" values supported for the Wallet Attestation (the `oauth-client-attestation+jwt`). Values MUST be chosen from Section *Cryptographic Algorithms* (page 615) and MUST NOT include `none` nor symmetric (MAC) algorithms.

continues on next page

Table 10.6 – continued from previous page

Claim	Description
client_attestation_pop_signing_alg_values_supported	JSON array containing the list of JWS "alg" values supported for the Wallet Attestation Proof-of-Possession (the `oauth-client-attestation-pop+jwt`). Values MUST be chosen from Section *Cryptographic Algorithms* (page 615) and MUST NOT include `none` nor symmetric (MAC) algorithms.
token_endpoint_auth_signing_alg_values_supported	JSON array containing a list of the signing algorithms ("*alg*" values) supported by the token endpoint for the signature on the JWT used to authenticate the client at the Token Endpoint. See **RFC 8414#section-2**.
require_signed_request_object	Boolean value. It MUST be set to *true* to indicate that the authorization request is protected using a signed Request Object [**RFC 9101**].
request_object_signing_alg_values_supported	JSON array containing a list of the signing algorithms ("*alg*" values) supported for Request Objects. See [openid-connect-discovery-1_0].
dpop_signing_alg_values_supported	JSON array containing a list of the signing algorithms ("*alg*" values) supported for DPoP proof JWTs. See **RFC 9449**.
jwks	JSON Web Key Set containing the cryptographic keys for the authorization server. See OID-FED Section 5.2.1 and JWK.

Important

If `token_endpoint_auth_methods_supported` includes `attest_jwt_client_auth`, the Authorization Server MUST include both `client_attestation_signing_alg_values_supported` and `client_attestation_pop_signing_alg_values_supported` in its metadata. Clients SHOULD fetch and parse the Authorization Server metadata to detect support and algorithm requirements for Attestation-Based Client Authentication. When algorithms are incompatible, the client MAY obtain a new client attestation using a supported algorithm.

Metadata for openid_credential_issuer

The *openid_credential_issuer* metadata contains the following claims.

Claim	Description
credential_issuer	REQUIRED. The Credential Issuer identifier. It MUST be a case sensitive URL using HTTPS scheme as defined in OpenID4VCI Sections 12.2.1 and 12.2.4.
credential_endpoint	REQUIRED. URL of the Credential endpoint. See OpenID4VCI Section 12.2.4.
nonce_endpoint	REQUIRED. URL of the Nonce Endpoint, as defined in Section 7 of OpenID4VCI.

continues on next page

Table 10.7 – continued from previous page

Claim	Description
deferred_credential_endpoint	OPTIONAL. URL of the deferred Credential endpoint, as defined in Section 12.2.4 of OpenID4VCI.
notification_endpoint	REQUIRED. It MUST be an HTTPs URL indicating the notification endpoint. See Section 12.2.4 of [OpenID4VCI].
authorization_servers	OPTIONAL. Array of strings, where each string is an identifier of the OAuth 2.0 Authorization Server (as defined in [**RFC 8414**]) the Credential Issuer relies on for authorization. If this parameter is omitted, the entity providing the Credential Issuer is also acting as the Authorization Server.
display	REQUIRED. See OpenID4VCI Section 12.2.4. Array of objects containing display language properties. The parameters that MUST be included are: • **name**: String value of a display name for the Credential Issuer. • **locale**: String value that identifies the language of this object represented as a language tag taken from values defined in *BCP47* **RFC 5646**. There MUST be only one object for each language identifier. • **logo**: Object with information about the logo of the Credential Issuer. The following parameters are included: – **uri**: REQUIRED. URL of the entity's logo that will be shown to the User during Wallet Instance interactions. The logo mime type MUST be `application/svg`. – **uri#integrity**: OPTIONAL. integrity metadata as defined in Section 3 of W3C-SRI. – **alt_text**: OPTIONAL. String value of the alternative text for the logo image.

continues on next page

Table 10.7 – continued from previous page

Claim	Description
credential_configurations_supported	REQUIRED. JSON object that outlines the details of the Digital Credentials supported by the Credential Issuer. It includes a list of name/value pairs, where each name uniquely identifies a specific supported Digital Credential. This identifier is utilized to inform the Wallet Instance which Digital Credential can be provided by the Credential Issuer. The associated value within the object MUST contain metadata specific to that Digital Credential, as defined following. See OpenID4VCI Sections 12.2.4 and A.3.2. • **format**: String identifying the format of this Credential. The Digital Credential MUST support the value string "*dc+sd-jwt*" in case of SD-JWT VC (See OpenID4VCI Section A.3.1.) and "*mso_mdoc*" in case of mdoc (see OpenID4VCI Section A.2.1.). • **scope**: JSON String identifying the supported *scope* value. The Wallet Instance MUST use this value in the Pushed Authorization Request. Scope values MUST be the entire set or a subset of the *scope* values in the *scopes_supported* parameter of the Authorization Server. If the Credential is included in the Digital Credentials Catalog the *scope* value MUST match with the `credential_type` parameter defined in *Digital Credentials Catalog Structure* (page 194) or in the *Schema Registry* (page 219). [See OpenID4VCI Section 12.2.4]. • **cryptographic_binding_methods_supported**: JSON Array of case sensitive strings that identify the representation of the cryptographic key material that the issued Credential is bound to. The Credential Issuer MUST support the value "*jwk*" for "dc+sd-jwt" format and "*cose_key*" for "mso_mdoc". • **credential_signing_alg_values_supported**: JSON Array of case sensitive strings that identify the algorithms that the Credential Issuer MUST support to sign the issued Credential. See Section *Cryptographic Algorithms* (page 615) for more details. • **proof_types_supported**: JSON object which provides detailed information about the key proof(s) supported by the Credential Issuer. It consists of a list of name/value pairs, where each name uniquely identifies a supported proof type. The Credential Issuer MUST support at least "*jwt*" as defined in OpenID4VCI Appendix F.1. The value associated with each name/value pair is a JSON object containing metadata related to the key proof(s). The Credential Issuer MUST support at least the parameter **proof_signing_alg_values_supported** which MUST be a JSON Array of case sensitive strings that identify the supported algorithms (see Section *Cryptographic Algorithms* (page 615) for more details about the supported algorithms). The Credential Issuer MAY support the parameter **key_attestations_required** as defined in Section 12.2 of OpenID4VCI. • **vct**: REQUIRED only if `format` is set to "*dc+sd-jwt*". As defined in [*SD-JWT-VC Credential Format* (page 358)]. • **doctype**: REQUIRED only if `format` is set to "*mso_mdoc*". As defined in [*mdoc-CBOR Credential Format* (page 366)]. • **credential_metadata**: REQUIRED. Object containing information relevant to the usage and display of issued Credentials. The parameters

Table 10.7 – continued from previous page

Claim	Description
jwks	REQUIRED. JSON Web Key Set document, passed by value, containing the protocol specific keys for the Credential Issuer. See OID-FED Section 5.2.1 and JWK.
trust_frameworks_supported	**REQUIRED. JSON array containing all supported trust frameworks. The supported values are:** • *it_cie*: CIE id trust framework supported. • *it_wallet*: IT-Wallet trust framework supported. • *eudi_wallet*: Member State EUDI Wallet trust framework supported. • *it_l2+document_proof*: eID Substantial Authentication with MRTD Verification protocol supported.
batch_credential_issuance	OPTIONAL. Object containing information about the Credential Issuer's support for issuance of Credentials in a batch at the Credential Endpoint. The presence of this parameter means that the Credential Issuer supports more than one key proof in the `proofs` parameter in the Credential request so can issue more than one Digital Credential for the same Credential with the same attributes about the Holder in a single request/response. The parameter that MUST be included is: • **batch_size**: Integer value specifying the maximum array size for the `proofs` parameter in a Credential request. The presence of this parameter advertises batch issuance only for the Credential types for which the Credential Issuer documents support. The Credential Issuer MUST declare whether PID and IT-Wallet ID are in scope of batch issuance. A Credential type without that advertisement MUST be issued as a single Credential (*CI_203* (page 776)).
status_list_aggregation_endpoint	REQUIRED. URL of the Status List Aggregation Endpoint. See TOKEN-STATUS-LIST Section 9.

10.3 Relying Party Solution

A Relying Party, as an Organizational entity relying upon IT-Wallet, provides Technical Solutions (Relying Party Solution) that could combine software, hardware, services, settings, and configurations, including Relying Party Instances for User authentication and Digital Credential verification.

In order to implement and provide Technical Solutions, a Relying Party may rely on the services of a **Relying Party Intermediary**. Pursuant to Article 5b(8) of EU_2024_1183, intermediaries acting on behalf of Relying Parties are deemed to be Relying Parties and **shall not store data about the content of the transaction** between the Wallet User and the intermediated Relying Party. For the complete definitions and the underlying European legal framework, see **Relying Party Intermediary** and **IT-Wallet Intermediary** in *Defined Terms and Acronyms* (page 639).

Note

In the implementation profile described by these technical specifications, a Relying Party Intermediary is also an OpenID Federation Intermediate Entity (**IT-Wallet Intermediary**).

A Relying Party provides at least one of the following components:

- **Relying Party Backend**: It handles Relying Party Instances registration and the relative Certificate management. It obtains X.509 Certificate according to the *The Infrastructure of Trust* (page 109). It SHOULD also provide a X.509 Certificate to its Relying Party Instances. It also may implement additional web services and business logic for its own purposes and use cases.
- **Relying Party Instance**: It is a frontend provided to Users in order to access Relying Party Services, and it may be provided as:
  - **Web Client**.
  - **Mobile Application**.

The following diagram depicts the Relying Party Solution High Level Architecture.

10.3.1 Relying Party Instance

The Relying Party Instance (RPI) is a mobile or embedded application designed to request, receive, and process Digital Credentials from Wallet Instances in a trusted manner. Each RPI ensures the integrity, confidentiality and authenticity of Credential exchanges, enabling secure interactions between Users and Relying Parties.

There are two primary types of Relying Party Instances, each serving different operational environments:

- **Embedded Relying Party Instance**: an hardware/software solution running on a specialized device (e.g., turnstiles). Each instance corresponds to a specific installation of the application on an embedded device. Since the trust with the Embedded Relying Party Instance is established through the trust established with the Mobile Relying Party Provider, the Embedded Relying Party Instance can be considered a confidential client.

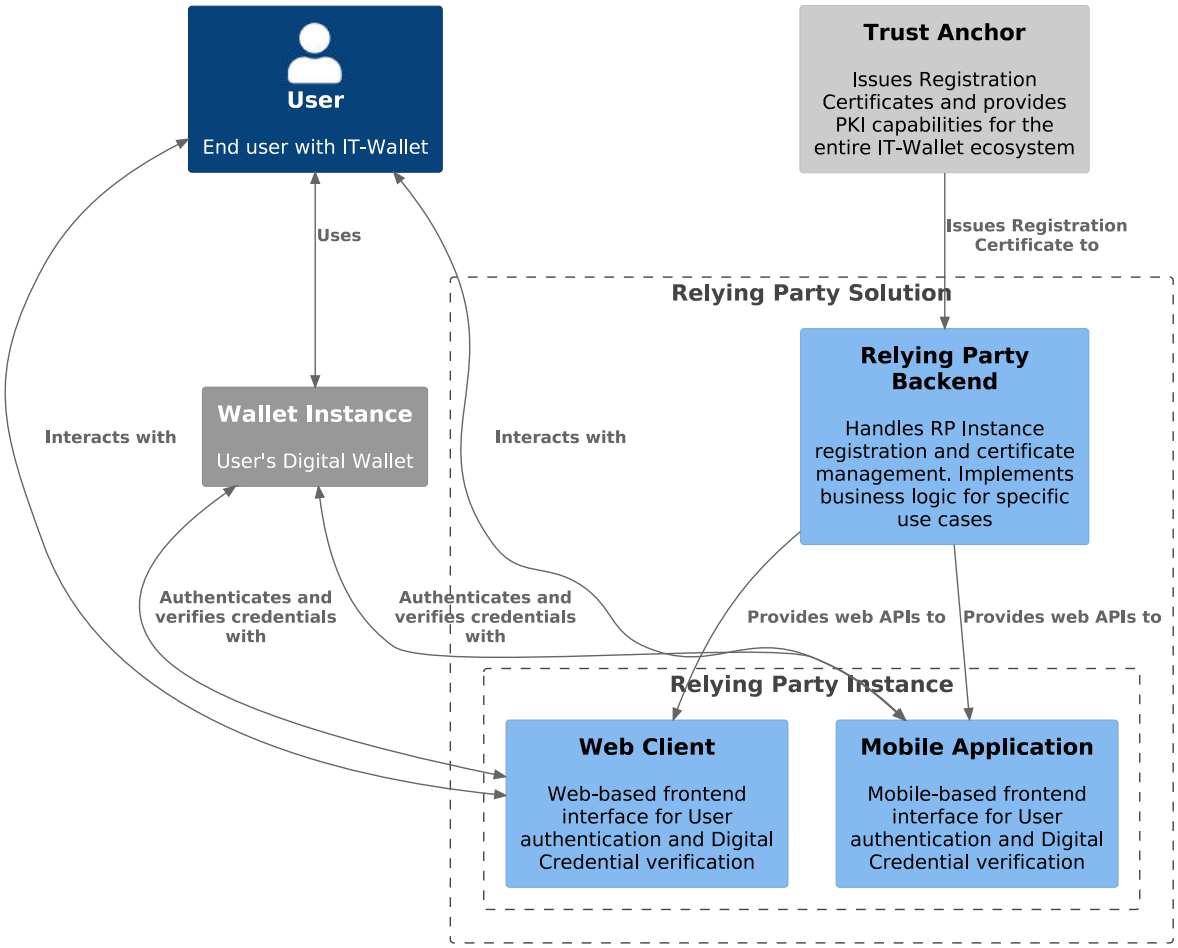


Fig. 10.11: Relying Party Solution High Level Architecture.

- **Mobile Relying Party Instance**: a native application running on a mobile device (e.g. , smartphone or tablet). Each instance corresponds to a specific installation of the application on a device. Since the trust with the Mobile Relying Party Instance is established through the trust established with the Mobile Relying Party Provider, the Mobile Relying Party Instance can be considered a confidential client.
- **Web Relying Party Instance**: a remote application operated by the Relying Party. A Web Relying Party Instance operates as a confidential client, meaning it can securely store confidential configurations (such as its private cryptographic keys) on a remote server. In this context, the Relying Party is unprovisioned and the presentation request and validation is automatically handled by software.

Note

Mobile and Embedded Relying Party Instances acting as public clients are not considered within these specifications.

Note

Unlike the Web Relying Party Instance, a Mobile and Embedded Relying Party Instance requires proper lifecycle management and special registration procedures managed through the Relying Party Backend.

Further technical and operational details are discussed in the following sections.

10.3.2 Mobile Relying Party Instance

A Mobile Relying Party Instance's lifecycle includes four main states: **Installed**, **Unverified**, **Verified**, and **Uninstalled**, supporting functionalities such as registration, Access Certificate reissuance, and revocation.

Mobile Relying Party Instance Lifecycle

In this section, state machines are presented to explain the Mobile Relying Party Instance states, as well as their transitions and relations.

As shown in Fig. 10.12, the Mobile Relying Party Instance has four distinct states: **Installed**, **Unverified**, **Verified**, and **Uninstalled**. Each state represents a specific functional status and determines the actions that can be performed.

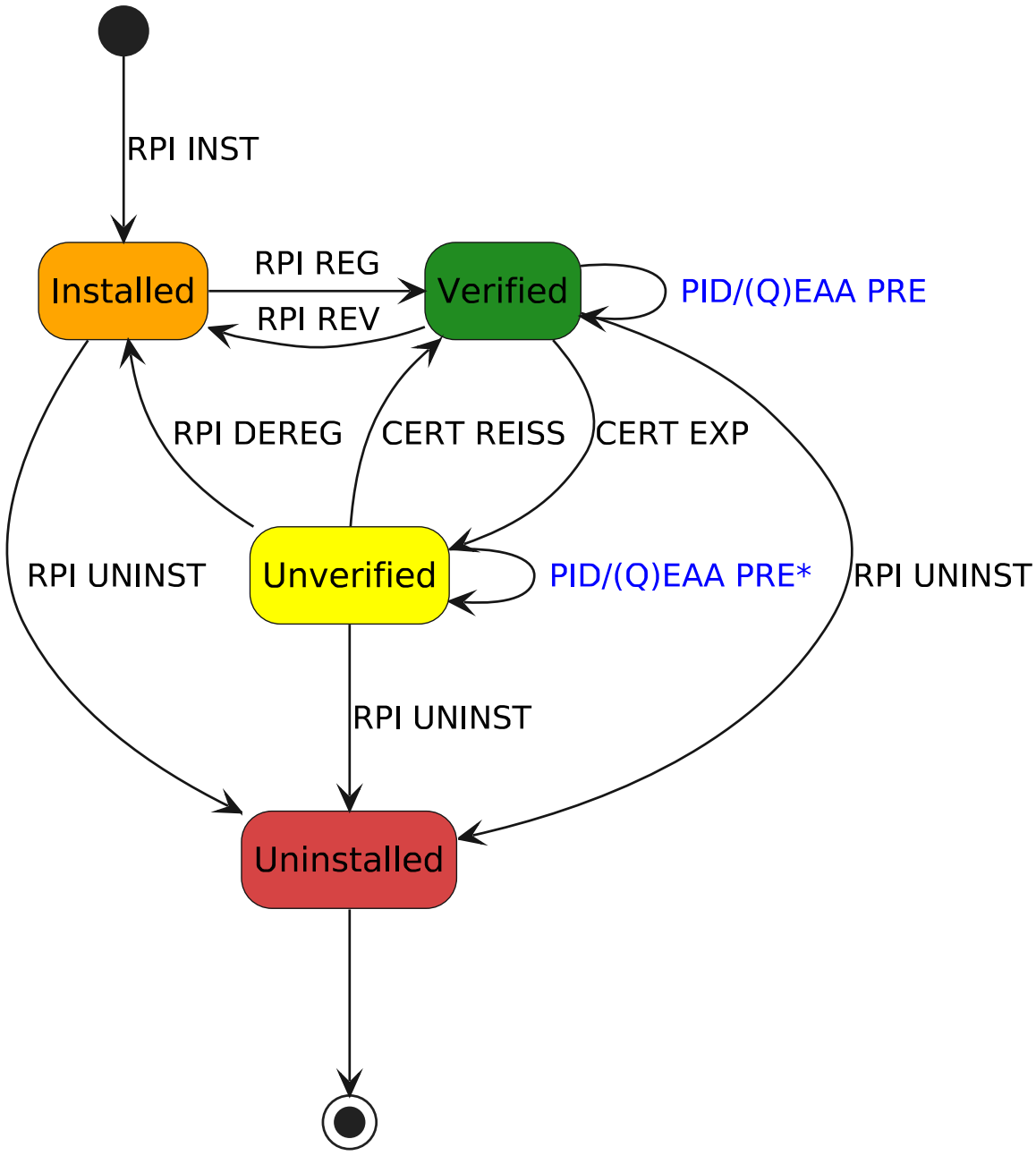


Fig. 10.12: Lifecycle of the Mobile Relying Party Instance.

Transition to Installed

The state machine begins with the Relying Party Instance installation (**RPI INST** transition), where Users download and install a Relying Party Instance using the official app store of their device's operating system, leading to the **Installed** state.

While in this state, the Relying Party Instance MUST interact only with the Relying Party Backend to be registered (i.e., to verify the Instance integrity, register Hardware Cryptographic Keys and obtain an Access Certificate).

When the revocation of the Relying Party Instance occurs (**RPI REV** transition), the Relying Party Instance MUST get back from **Verified** to **Installed**. This transition implies the following operations:

1. The Access Certificate MUST be revoked.
2. The Hardware Cryptographic Keys MUST be deleted.

Revocation can occur in the following cases:

- For security reasons (e.g., compromise of cryptographic material).
- For technical reasons (e.g., deprecation of the Relying Party Solution).
- In case of Relying Party de-registration (as detailed in EIDAS-ARF, Section 6.4.3).
- Illegal activities reported by Judicial or Supervisory Bodies.

In addition, each Relying Party SHOULD set an amount of time (grace period) during which the Relying Party Instance can request presentations of Digital Credentials by authenticating itself towards a Wallet Instance using an expired Access Certificate. After this period, the Relying Party Instance MUST be de-registered (**RPI DEREG** transition) and go back to the **Installed** state. This transition implies that the Hardware Cryptographic Keys MUST be deleted.

Transition to Verified

The Relying Party Instance needs to obtain a proper Access Certificate, which will be used to authenticate itself towards Wallet Instances. This Certificate is obtained by interacting with the Relying Party Backend, which in turns communicates with the Relying Party Instance Access Certificate Authority. Specifically, the registration transition (**RPI REG**) consists of the following subphases, leading to the **Verified** state:

1. **Initialization**: After verification of the Relying Party Instance integrity, it registers a pair of Hardware Cryptographic Keys.
2. **Access Certificate Issuance**: The Relying Party Instance obtains an Access Certificate.

In case the Access Certificate is expired, a new Certificate can be issued to the Relying Party Instance; this operation is represented by the **CERT REISS** transition towards the **Verified** state.

While in this state, the Relying Party Instance can request the presentation of Digital Credentials to Wallet Instances (**PID/(Q)EAA PRE**), using the Access Certificate to authenticate itself.

Transition to Unverified

The expiration of the Access Certificate (**CERT EXP** transition) leads to the **Unverified** state.

While in this state, the Relying Party Instance can still request the presentation of Digital Credentials to Wallet Instances during the grace period. However, as the Certificate is expired, a specific disclaimer MUST be displayed to the User of the Wallet Instance during the presentation flow; for this reason, this operation is represented by the label **PID/(Q)EAA PRE**. This is required to support offline presentation flows. After the grace period has passed, the Relying Party Instance MUST NOT longer request presentations and will be de-registered.

Transition to Uninstalled

Across the **Installed**, **Verified** and **Unverified** states, the Relying Party Instance can be removed entirely through the Relying Party Instance uninstall (**RPI UNINST**) transition, leading to the **Uninstalled** state. If a Relying Party Instance is **Uninstalled**, it ends its lifecycle.

Mobile Relying Party Instance Functionalities

A Mobile Relying Party Instance MUST support three fundamental functionalities: **Registration**, **Access Certificate Reissuance**, and **Revocation**. Each functionality is described in detail in the following sections.

Note

Throughout this section, the services used to attest genuineness of the Relying Party Instance and the device in which it is installed are referred to as **Device Integrity Service**. The Device Integrity Service is considered in an abstract fashion and it is assumed to be a service provided by a trusted third party (i.e., the OS Provider's API) which is able to perform integrity checks on the Wallet Instance as well as on the device where it is installed.

Note

The details provided below are non-normative and are intended to clarify the functionalities of the Mobile Relying Party Instance. The actual implementation may vary based on the specific use case and requirements of the Relying Party.

Mobile Relying Party Instance Registration

This process allows for the registration of a Relying Party Instance with the Relying Party Backend, and the issuance of an Access Certificate that will be used for authentication purposes towards Wallet Instances during presentation flows. The process consists of two subphases:

1. **Initialization**: The Relying Party Instance requests a security and integrity assertion from the OS manufacturer, which also binds a long-lived key pair stored in a proper secure storage within the device itself; the assertion is then validated by the Relying Party Backend. Further details are provided in Section *Mobile Application Instance* (page 667).
2. **Access Certificate Issuance**: The Relying Party Instance requests an Access Certificate from the Relying Party Backend. Before interacting with the Relying Party Instance Access Certificate Authority for the issuance of the Access Certificate, the Relying Party Backend validates the Relying Party Instance's integrity and security by leveraging the long-lived, attested keys generated in the previous subphase. The flow is displayed in Fig. 10.13, while a step-by-step description is provided below.

Note

Access Certificates MAY be issued as short-lived (typically valid within 24 hours) or long-lived.

Steps 1-2: The Mobile Relying Party Instance:

1. Verifies the existence of Cryptographic Hardware Keys. If none exist, the Relying Party Instance re-initialization is required.
2. Generates an asymmetric key pair for the Access Certificate (`key_pub`, `key_priv`).

Steps 3-5: The Mobile Relying Party Instance requests a `nonce` from the relying-party-provider-backend-endpoint:Relying Party Provider Backend Nonce Endpoint of the Relying Party Backend. This `nonce` MUST be unpredictable to serve as the main defense against replay attacks.

Upon a successful request, the relying-party-provider-backend-endpoint:Relying Party Provider Backend Nonce Endpoint generates and returns the `nonce` to the Mobile Relying Party Instance. The relying-party-provider-backend-endpoint:Relying Party Provider Backend Nonce Endpoint MUST ensure that it is single-use and valid only within a specific time frame.

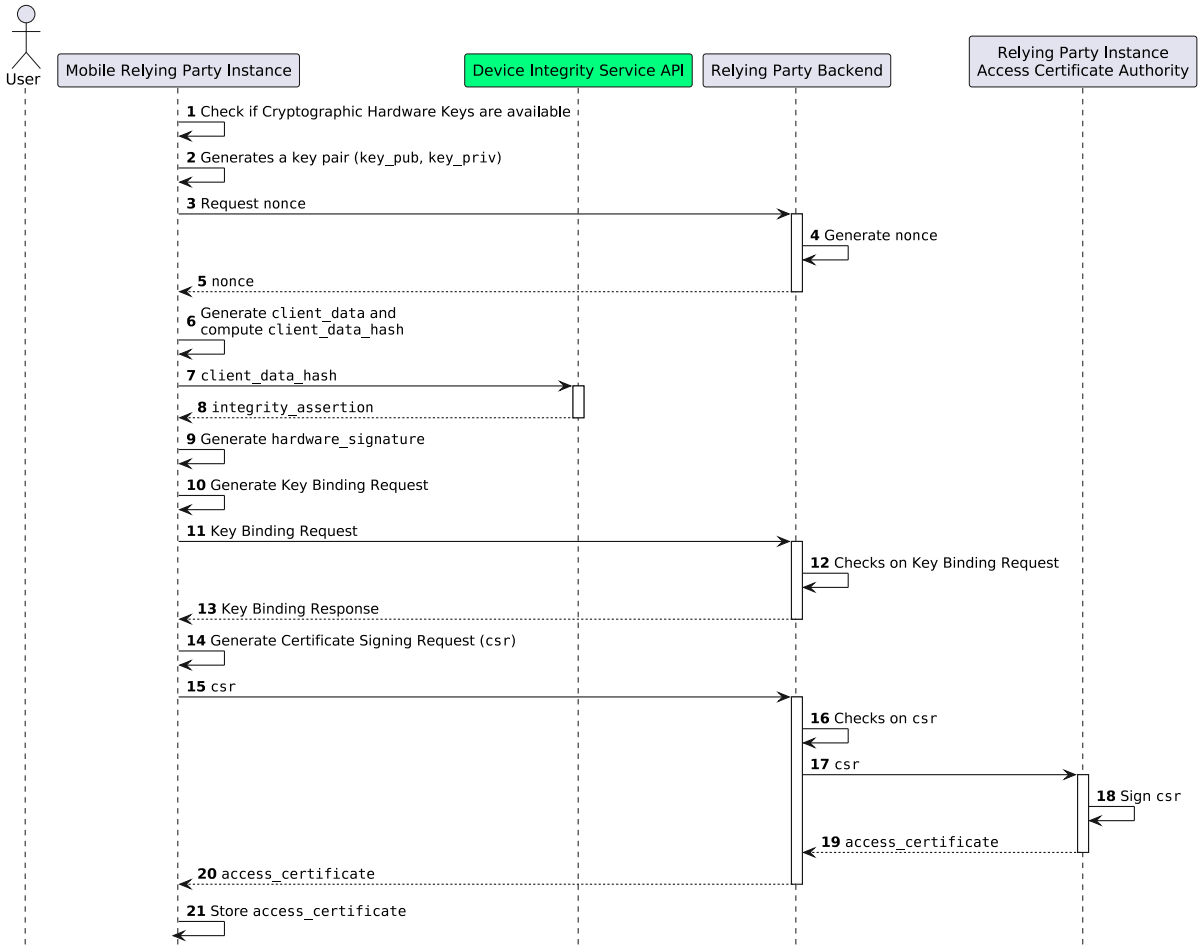


Fig. 10.13: Flow of the Mobile Relying Party Instance Access Certificate Issuance.

Non-normative examples of the Nonce Request and Response can be found in the *Mobile Application Nonce Request* (page 672) and *Mobile Application Nonce Response* (page 672) sections, respectively.

Step 6: The Mobile Relying Party Instance:

1. Generates `client_data`, a JSON object that includes the challenge and the thumbprint of `key_pub`, obtained from its `JWK` representation.
2. Computes `client_data_hash` by applying the SHA256 algorithm to `client_data`.

Below is a non-normative example of the `client_data` JSON object.

```
{
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
  "jwk_thumbprint": "hT3v7KQjFZy6GvDkYgOZ1u2F6T4Nz5bPjX8o1MZ3dJY"
}
```

Steps 7-8: The Mobile Relying Party Instance:

1. Requests the Device Integrity Service to create a `integrity_assertion` value linked to the `client_data_hash`.
2. Receives a signed `integrity_assertion` value from the Device Integrity Service, authenticated by the OEM.

Steps 9-11: The Mobile Relying Party Instance:

1. Generates an `hardware_signature` value by signing the `client_data_hash` with the Hardware Cryptographic private key, serving as a proof of possession for the Cryptographic Hardware Keys.
2. Generates the relying-party-provider-backend-endpoint:Relying Party Provider Backend Key Binding Request in the form of a JWT. This JWT includes `integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag`, and `cnf` (representing `key_pub`); it is signed using `key_priv`.
3. Sends the signed relying-party-provider-backend-endpoint:Relying Party Provider Backend Key Binding Request JWT as an `assertion` parameter in the body of an HTTP request to the relying-party-provider-backend-endpoint:Relying Party Provider Backend Key Binding Endpoint.

Step 12: The Relying Party Backend evaluates the Key Binding Request and performs the following checks:

1. The request includes all required HTTP header parameters as defined in relying-party-provider-backend-endpoint:Relying Party Provider Backend Key Binding Request.
2. The signature of the Key Binding Request is valid and verifiable using the provided `jwk`.

3. The `nonce` value has been generated by the Relying Party Backend and not previously used.
4. The Relying Party Instance has valid Cryptographic Hardware Keys registered.
5. The `client_data` can be reconstructed using `nonce` and `cnf` (representing `key_pub`). The `hardware_signature` parameter value is then validated using the registered Cryptographic Hardware Key's public key associated with the Relying Party Instance.
6. The `integrity_assertion` can be validated according to the device manufacturer's guidelines. The specific checks performed by the Relying Party Backend are detailed in the operating system manufacturer's documentation.
7. The device in use is free of known security flaws and meets the minimum security requirements defined by the Relying Party.
8. The URL in the `iss` parameter matches the Relying Party's URL identifier.

Step 13: If the checks are successful, the Relying Party Backend responds with a confirmation of success (relying-party-provider-backend-endpoint:Relying Party Provider Backend Key Binding Response).

Step 14: The Mobile Relying Party Instance generates a Certificate Signing Request (CSR, `csr`) using `key_pub` and `key_priv`.

Step 15: The Mobile Relying Party Instance sends the CSR to the relying-party-provider-backend-endpoint:Relying Party Provider Backend Access Certificate Endpoint of the Relying Party Backend, as part of the relying-party-provider-backend-endpoint:Relying Party Provider Backend Access Certificate Request.

Steps 16-17: The Relying Party Backend checks that the public key in the CSR corresponds to a Relying Party Instance that has been previously validated, i.e., that it matches the one bound to the Cryptographic Hardware Keys through `hardware_signature`. If this check is successful, the Relying Party Backend sends the CSR to the Relying Party Instance Access Certificate Authority.

Steps 18-19: The Relying Party Instance Access Certificate Authority signs the CSR, obtaining a valid Access Certificate (`access_certificate`) which it sends back to the Relying Party Backend.

Steps 20-21: The Relying Party Backend sends the Access Certificate (as part of the relying-party-provider-backend-endpoint:Relying Party Provider Backend Access Certificate Response) to the Mobile Relying Party Instance, which stores it for future authentication towards Wallet Instances.

Mobile Relying Party Instance Access Certificate Reissuance

The issuance of a new Access Certificate follows the same flow described in the *Mobile Relying Party Instance Registration* (page 347) section for **Access Certificate Issuance**. Those certificates MAY be issued as short-lived (typically valid within 24 hours) or long-lived.

Mobile Relying Party Instance Revocation

Relying Parties MUST periodically verify the Relying Party Instance's authenticity and security. When security issues are detected, Relying Parties MUST revoke the Relying Party Instance, revoking its X.509 Access Certificate (in case of long-lived certificates), and in any case, Relying Parties MUST NOT allow the re-issue of certificates. As a result, Mobile Relying Party Instance revocation MUST be tied to X.509 Access Certificates validity.

Long-lived X.509 Certificates follows the requirements about their lifecycle, defined in *The Infrastructure of Trust* (page 109).

10.3.3 Web Relying Party Instance

Web Instances operates server-side security controls that safely store secrets and cryptographic keys in a controlled environment. Web Instances MUST be registered with the Trust Anchor or Intermediary Entities, according to *The Infrastructure of Trust* (page 109).

Web Relying Party Instance Functionalities

A Web Relying Party Instance MUST support two fundamental functionalities: **Registration** and **Revocation**. Each functionality is described in the following sections.

Web Relying Party Instance Registration

Web Relying Party Instances, as confidential clients, are registered directly with the Trust Anchor or an Intermediary Entity. The registration involves:

- The Relying Party MUST register its Web Instance with the Trust Anchor or Intermediary.
- The Relying Party MUST expose an Entity Configuration as defined in the Trust Framework.
- The Entity Configuration MUST contain all necessary metadata for federation, including endpoints and public keys.
- No individual instance lifecycle management is required, as the Web Instance operates as part of the secured server environment.

Web Relying Party Instance Revocation

When a Web Relying Party Instance needs to be revoked:

- The revocation MUST be performed according to the Trust Framework procedures.
- The cryptographic keys used by the Web Instance MUST be revoked.
- The Entity Configuration MUST be updated to reflect the revocation.
- The Trust Anchor MUST be notified of the revocation to update federation metadata.

10.3.4 Relying Party Entity Configuration

According to Section *Configuration of the Federation* (page 116), as a Federation Entity, the Relying Party is required to maintain a well-known endpoint that hosts its Entity Configuration. The Entity Configuration of Relying Parties MUST contain the parameters defined in the Sections *Entity Configuration Leaves and Intermediates* (page 120) and *Entity Configurations Common Parameters* (page 118).

The Relying Parties MUST provide the following metadata types:

- *federation_entity*
- *openid_credential_verifier*

The *federation_entity* metadata MUST contain the claims as defined in Section *Metadata of federation_entity Leaves* (page 122).

Example of a Relying Party Entity Configuration

Below a non-normative example of the request made by the Wallet Instance to the *openid-federation* well-known endpoint to obtain the Relying Party Entity Configuration:

```
GET /.well-known/openid-federation HTTP/1.1
HOST: relying-party.example.org
```

Below is a non-normative response example:

```
{
    "iat": 1718207217,
    "exp": 1749743216,
    "iss": "https://relying-party.example.org",
    "sub": "https://relying-party.example.org",
    "authority_hints": [
        "https://trust-anchor.example.org"
    ],
    "jwks": {
```

```
        "keys": [
            {
                "kid": "FANFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
            }
        ]
    },
    "metadata": {
        "federation_entity": {
            "homepage_uri": "https://relying-party.example.org",
            "organization_name": "Organization Name",
            "contacts": [
                "informazioni@example.it",
                "protocollo@pec.example.it"
            ],
            "tos_uri": "https://relying-party.example.org/public/info_policy.html",
            "policy_uri": "https://relying-party.example.org/public/privacy_policy.html",
            "logo_uri": "https://relying-party.example.org/public/logo.svg"
        },
        "openid_credential_verifier": {
            "application_type": "web",
            "client_id": "https://relying-party.example.org",
            "client_name": "Organization Name",
            "logo_uri": "https://relying-party.example.org/public/compact-logo.svg",
            "contacts": [
                "informazioni@example.it",
                "protocollo@pec.example.it"
            ],
            "request_uris": [
                "https://relying-party.example.org/request_uri"
            ],
            "response_uris": [
                "https://relying-party.example.org/response_uri"
            ],
            "redirect_uris": [
                "https://relying-party.example.org/cb"
            ],
            "encrypted_response_enc_values_supported": [
                "A256GCM"
            ],
            "vp_formats_supported": {
                "dc+sd-jwt": {
                    "sd-jwt_alg_values": [
                        "ES256",
```

```
                    "ES384",
                    "ES512"
                ],
                "kb-jwt_alg_values": [
                    "ES256"
                ]
            },
            "mso_mdoc": {
                "issuerauth_alg_values": [
                    -9,
                    -51
                ],
                "deviceauth_alg_values": [
                    -9,
                    -51
                ]
            }
        },
        "jwks": {
            "keys": [
                {
                    "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                    "kty": "EC",
                    "crv": "P-256",
                    "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                    "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                }
            ]
        }
    }
}
```


10.3.5 Relying Party Metadata

The *openid_credential_verifier* metadata MUST contain the *client_metadata*, as included in the parameters shown below. (test-plans-remote-presentation:Remote Credential Verifier Test Matrix). Please note that *openid_credential_verifier* is a OpenID Federation specific metadata used for OpenID4VP and it is used when the Relying Party's `client_id` is set with `openid_federation`. When the Relying Party's `client_id` parameter is set with `x509_hash`, the metadata is instead conveyed in the `client_metadata` parameter provided within the request.

Claim	Description
client_id	It MUST contain an HTTPS URL that uniquely identifies the RP. See **RFC 7591#section-3.2.1** and OpenID Connect Dynamic Client Registration 1.0 Section 3.2.
client_name	Human-readable string name of the RP. See **RFC 7591#section-2**.
logo_uri	URL of the entity's logo that will be shown to the User during interactions with the Wallet Instance. See OID-FED Section 5.2.2. The logo mime type MUST be `application/svg`.
application_type	String indicating the type of application. It MUST be set to "*web*" value. See OpenID Connect Dynamic Client Registration 1.0 Section 2.
request_uris	JSON Array of *request_uri* values that are pre-registered by the RP. These URLs MUST use the *https* scheme. See OpenID Connect Dynamic Client Registration 1.0 Section 2.
response_uris	JSON Array of response URI strings to which the Wallet Instance MUST send the Authorization Response using an HTTP POST request as defined by the Response Mode `direct_post` and `direct_post.jwt` (see OpenID4VP Draft 20 Sections 6.2 and 6.3).
encrypted_response_enc_values_supported	JSON array of content encryption algorithms ("enc") the Verifier supports for encrypting the authorization response when using response mode `direct_post.jwt`. See OpenID4VP §5.1 and §8.3.1.
vp_formats_supported	JSON object defining the formats and proof types of Verifiable Presentations and Verifiable Credentials the RP supports. It consists of a list of name/value pairs, where each name uniquely identifies a supported type. The RP MUST support at least `dc+sd-jwt`. For SD-JWT VC, the value associated with each name/value pair MUST include `sd-jwt_alg_values` listing acceptable signing algorithms; for mdoc-CBOR, the value MUST include `issuerauth_alg_values` and `deviceauth_alg_values`. The JOSE/COSE headers of presented artifacts MUST match one of the advertised values. See OpenID4VP §11 and Appendix B.
jwks	JSON Web Key Set document, passed by value, containing the protocol specific keys for the Relying Party. See OID-FED Section 5.2.1 and JWK.
erasure_endpoint	[CONDITIONAL] JSON String that represents the URI to which the Wallet Instance can request deletion of Users' attributes. This URL MUST use the *https* scheme. Upon receiving an erasure request, the Relying Party MUST uniquely identify one or more Digital Credentials for which the User requests deletion, by applying identity matching as specified in *Identity Matching and Identity Reconciliation* (page 538).

Note

The parameters **response_uris** and **erasure_endpoint** are introduced in this specification.

10.4 Authentic Sources

Authentic Sources provide Users attributes to the Credential Issuers enabling them in the issuance of the Digital Credentials. During the Issuance Flow, Credential Issuers request from Authentic Sources the attributes required to provide the requested Credential. Authentic Sources MAY also provide a Credential Offer related to their Credential Issuers as defined in Section *Credential Offer Flow* (page 452).

Public Authentic Sources MUST interact with Credential Issuers via PDND according to the rules defined in Section *e-Service PDND* (page 675) and in Section *Status Update by Authentic Sources* (page 408). See also Section *e-Service PDND Authentic Source Catalog* (page 602) for additional details.

Authentic Sources MUST:

- provide User's attributes when requested by the Credential Issuer authorized to issue the related Digital Credential attesting the attributes. Public Authentic Sources MUST use PDND to send User's attributes to their Credential Issuers. When the User's attributes is not available during the Issuance Flow, Authentic Sources MUST provide Credential Issuers with an estimated time when the User's data will be available. Authentic Sources MAY require an evidence that:
  - the request for Users attributes is related to data about themselves;
  - the request for User attributes comes from a valid Wallet Instance;
- cooperate with their Credential Issuers so that the attributes attested in a Digital Credential are always kept up to date. Public Authentic Sources MUST use PDND Signal Hub to notify their Credential Issuers of any update regarding attributes that have changed or are no longer valid.

CHAPTER

ELEVEN

DIGITAL CREDENTIAL MANAGEMENT

11.1 Digital Credential Data Model

A Digital Credential data model has the following structure:

- **Metadata attributes**:
  - **Format-Agnostic**: These are high-level metadata attributes that describe the Digital Credential independently of its encoding format. They represent the semantic information about the Credential (e.g., `credential_type_identifier`, `issuing_authority`, `expiry_date`) and remain conceptually consistent across different formats. When a Credential is encoded, these common metadata attributes are mapped to format-specific technical parameters according to the encoding rules of each format (SD-JWT-VC or mdoc-CBOR).
  - **Format-Specific**: These are format-specific metadata parameters that support the security model and protocol requirements.
- **User attributes**: Information about the User, such as identity or qualifications.

The (Q)EAAs are issued by (Q)EAA Issuers to a Wallet Instance and MUST be provided in SD-JWT VC or mdoc-CBOR data format. The (Q)EAA data model is use-case driven and may include different User attributes according to its specific purpose. The (Q)EAA metadata attributes are specific for each data format, as described in the following sections.

11.1.1 Format-Agnostic Credential Metadata Attributes

The following table defines the common metadata attributes that are applicable to Digital Credentials regardless of their encoding format. These attributes represent the semantic information about the Credential.

Data Identifier	Description
credential_type_identifier	REQUIRED. A unique and collision-resistant identifier that specifies the type and schema of the Digital Credential. It defines the set of claims/attributes that the Digital Credential contains and their structure.
issuing_authority	REQUIRED. Name of the administrative authority that issued the Digital Credential.
issuing_country	REQUIRED. Alpha-2 country code, as specified in ISO 3166-1, of the country or territory of the Credential Issuer.
issuance_date	OPTIONAL. Date (and if possible time) when the Digital Credential was issued and/or the administrative validity period of the Digital Credential began.
expiry_date	OPTIONAL. Date (and if possible time) when the Digital Credential will expire.
location_status	OPTIONAL. The location of validity status information on the Digital Credential where the Credential Issuer revoke Digital Credential.
cryptographic_binding	OPTIONAL. Object containing the proof-of-possession key materials.
verification	OPTIONAL. Object containing Identity proofing and User data verification information.

The following sections provide format-specific attributes and a mapping of the above metadata attributes to format-specific technical parameters when the credential is encoded in SD-JWT VC or mdoc-CBOR format.

11.1.2 SD-JWT-VC Credential Format

When Digital Credentials are issued in the SD-JWT VC format, they MUST be compliant to SD-JWT and SD-JWT-VC specifications.

SD-JWT-VC Digital Credentials MUST be signed using the Issuer's private key. SD-JWT VC Digital Credentials MAY be provided along with a Type Metadata Document related to the issued Credential according to Sections 6 and 6.3 of [SD-JWT-VC]. The payload of Digital Credentials MUST contain the **_sd_alg** claim described in Section 4.1.1 SD-JWT and other claims specified in this section.

The claim **_sd_alg** indicates the hash algorithm used by the Issuer to generate the digests as described in Section 4.1.1 of SD-JWT. **_sd_alg** MUST be set to one of the specified algorithms in Section *Cryptographic Algorithms* (page 615).

Claims that are not selectively disclosable MUST be included in the SD-JWT as they are. The digests of the disclosures, along with any decoy if present, MUST be contained in the **_sd** array, as specified in Section 4.2.4.1 of SD-JWT.

The Disclosures are provided to the Holder together with the SD-JWT in the *Combined Format for Issuance* that is an ordered series of base64url-encoded values, each separated from the next by a single tilde ('~') character as follows:

```
<Issuer-Signed-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure N>~
```

See SD-JWT-VC and SD-JWT for additional details.

Digital Credential SD-JWT Metadata Attributes

The JOSE header contains the following mandatory parameters:

Claim	Description	Reference
typ	REQUIRED. It MUST be set to `dc+sd-jwt` as defined in SD-JWT-VC.	**RFC 7515** Section 4.1.9.
alg	REQUIRED. Signature Algorithm.	**RFC 7515** Section 4.1.1.
kid	REQUIRED. Unique identifier of the public key.	**RFC 7515** Section 4.1.8.
trust_chain	OPTIONAL. JSON array containing the trust chain that proves the reliability of the issuer of the JWT.	[OID-FED] Section 4.3.
x5c	REQUIRED. Contains the X.509 public key certificate or certificate chain [**RFC 5280**] corresponding to the key used to digitally sign the JWT.	**RFC 7515** Section 4.1.8 and [SD-JWT-VC] Section 3.5.

The JWT payload contains the following claims. Unless otherwise specified, the following claims MUST NOT be selectively disclosable.

Claim	Description	Reference
iss	REQUIRED. *String*. URL string representing the Credential Issuer unique identifier.	[RFC7519, Section 4.1.1].
sub	OPTIONAL. *String*. The identifier of the subject of the Digital Credential, the User, MUST be opaque and MUST NOT correspond to any anagraphic data or be derived from the User's anagraphic data via pseudonymization. Additionally, it is required that two different Credentials issued MUST NOT use the same `sub` value.	[RFC7519, Section 4.1.2].
iat	OPTIONAL. UNIX Timestamp with the time of JWT issuance, coded as NumericDate as indicated in **RFC 7519**.	[RFC7519, Section 4.1.6].

continues on next page

Table 11.3 – continued from previous page

Claim	Description	Reference
exp	REQUIRED. UNIX Timestamp with the expiry time of the JWT, coded as NumericDate as indicated in **RFC 7519**. In accordance with [EIDAS-ARF] HLR **ISSU_12c** and **ISSU_12d** it MUST NOT be later than the expiration date of the Wallet Unit Attestation presented as part of the Digital Credential issuance process.	[RFC7519, Section 4.1.4].
nbf	OPTIONAL. UNIX Timestamp with the start time of validity of the JWT, coded as NumericDate as indicated in **RFC 7519**.	[RFC7519, Section 4.1.4].
issuing_authority	REQUIRED. *String*. Format-encoded data identifier *issuing_authority* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357).	Commission Implementing Regulation EU_2024/2977.
issuing_country	REQUIRED. *String*. Format-encoded data identifier *issuing_country* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357).	Commission Implementing Regulation EU_2024/2977.
issuance_date	OPTIONAL. *String*. Format-encoded data identifier *issuance_date* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). This attribute pertains to the administrative issuance date, which is typically different from the technical issuance date expressed by the JWT `iat` claim.	Section 2.6 of the ARF PID Rulebook v1.3 [EIDAS-ARF].
date_of_expiry	OPTIONAL. *String*. Format-encoded data identifier *expiry_date* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). This attribute pertains to the administrative validity period of the Digital Credential, which is typically different from the technical validity period expressed by the JWT `exp` claim.	Commission Implementing Regulation EU_2024/2977.
status	OPTIONAL. REQUIRED only if the Digital Credential is long-lived. *JSON object*. Format-encoded data identifier *location_status* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). It MUST contain either the JSON member *status_list*.	Section 3.2.2.2 SD-JWT-VC.
cnf	OPTIONAL. *JSON object*. Format-encoded data identifier *cryptographic_binding* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357), containing the proof-of-possession key materials. By including a **cnf** (confirmation) claim in a JWT, the Issuer of the JWT declares that the Holder is in control of the private key related to the public one defined in the **cnf** parameter. The recipient MUST cryptographically verify that the Holder is in control of that key.	[RFC7800, Section 3.1] and Section 3.2.2.2 SD-JWT-VC.

continues on next page

Table 11.3 – continued from previous page

Claim	Description	Reference
vct	REQUIRED. *String*. Format-encoded data identifier *credential_type_identifier* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). Credential type value MUST be a URN and it MUST be set using one of the values obtained from the Credential Issuer metadata, matching of the literals included in this URN MUST be performed in a case-sensitive manner. It is the identifier of the SD-JWT VC type and it MUST be set with a collision-resistant value as defined in Section 2 of **RFC 7515**. It MUST contain also the number of version of the Credential type. Unless otherwise specified by EIDAS-ARF and EUDI Rulebooks, the *vct* SHOULD follow a structure like *urn:it-wallet:{credential_type}:{credential_type_version}*.	Section 3.2.2.2 SD-JWT-VC.
vct#integrity	OPTIONAL. *String*. The value MUST be an "integrity metadata" string as defined in Section 3 of [W3C-SRI]. *SHA-256*, *SHA-384* and *SHA-512* MUST be supported as cryptographic hash functions. *MD5* and *SHA-1* MUST NOT be used. This claim MUST be verified according to Section 3.3.5 of [W3C-SRI].	Section 6.1 SD-JWT-VC, [W3C-SRI]
verification	OPTIONAL. *JSON object*. Format-encoded data identifier *verification* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). It includes the following sub-value: • `trust_framework`: REQUIRED. *String* identifying the trust framework used for User authentication. It MUST be set using one of the values described in the *trust_frameworks_supported* map provided within the Credential Issuer Metadata. • `assurance_level`: REQUIRED. *String* identifying the level of identity assurance guaranteed during the User authentication process. The value MUST match with one of the values mapped in the `acr_values_supported` array of the credential-issuer-metadata:Credential Issuer Metadata.	Domestic extension.
__sd	REQUIRED. *Array of strings*, where each string represents a digest of a Disclosure.	4.2.4.1 SD-JWT
__sd__alg	REQUIRED. *String*. Hash algorithm used by the Issuer to generate the digests.	4.1.1 SD-JWT

Note

The standard JWT claims `nbf` and `exp` are used to express the technical validity period of a SD-JWT VC-compliant Digital Credential.

The `status` parameter `status_list`, it MUST be a *JSON object* compliant with Section 6.2 of TOKEN-STATUS-LIST.

(Q)EAA non-normative Examples

Below is a non-normative example of (Q)EAA in JSON.

```
{
    "iss": "https://issuer.example.org",
    "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
    "iat": 1683000000,
    "exp": 1883000000,
    "issuing_authority": "Credential Issuer Organization",
    "issuing_country": "IT",
    "date_of_expiry": "2033-03-19",
    "status": {
        "status_list": {
            "idx": 5678,
            "uri": "https://issuer.example.org/status"
        }
    },
    "vct": "urn:it-wallet:disabilitycard:1",
    "vct#integrity": "2e40bcd6799008085ffb1a1f3517efee335298fd976b3e655bfb3f4eaa11d171",
    "verification": {
        "trust_framework": "it_wallet",
        "assurance_level": "https://trust-anchor.example.it/loa/high"
    },
    "document_number": "XXXXXXXXXX",
    "given_name": "Mario",
    "family_name": "Rossi",
    "birth_date": "1980-01-10",
    "tax_id_code": "TINIT-XXXXXXXXXXXXXXXX",
    "constant_attendance_allowance": true
}
```

The corresponding SD-JWT for the previous data is represented as follow, as decoded JSON for both header and payload.

```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
```

```
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
    "_sd": [
        "D4VkWjnA0WON7HdCGFtU869MSvORHPf8p5fQRD5gNj0",
        "JOQk0kuBSVk80rFlv9VGY-yiIzsfzEJKk3d4RROfzkM",
        "Q7TX7kL8CNUp3BFBKP5xxIuPu5gRgkO6HplM3E1iMIc",
        "Wq3gFfmC0I9Lefw1mh-Bk5XPRtoSCg9aE23uOhxakas",
        "oF2qeWAbKO_qWGQ5z-HGKeifl2PMIEMbJe8L-PJ-wko",
        "qbRtUHp9Oax9dm5GeKnw_W12Yu1E2DoU6wrFPee7aBo"
    ],
    "exp": 1883000000,
    "iss": "https://issuer.example.org",
    "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
    "iat": 1683000000,
    "issuing_authority": "Credential Issuer Organization",
    "issuing_country": "IT",
    "date_of_expiry": "2033-03-19",
    "status": {
        "status_list": {
            "idx": 5678,
            "uri": "https://issuer.example.org/status"
        }
    },
    "vct": "urn:it-wallet:disabilitycard:1",
    "vct#integrity": "2e40bcd6799008085ffb1a1f3517efee335298fd976b3e655bfb3f4eaa11d171",
    "verification": {
        "trust_framework": "it_wallet",
        "assurance_level": "https://trust-anchor.example.it/loa/high"
    },
    "_sd_alg": "sha-256",
    "cnf": {
        "jwk": {
            "kty": "EC",
            "crv": "P-256",
            "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
            "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
        }
    }
}
```


In the following the disclosure list is given:

Claim `document_number`:

- SHA-256 Hash: `D4VkWjnA0WON7HdCGFtU869MSvORHPf8p5fQRD5gNj0`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImRvY3VtZW50X251bWJlciIs ICJYWFhY-WFhYWFhYIl0`
- Contents: `["kghte5MDNHbQfdJHp88pCA", "document_number", "XXXXXXXXXX"]`

Claim `given_name`:

- SHA-256 Hash: `qbRtUHp9Oax9dm5GeKnw_W12Yu1E2DoU6wrFPee7aBo`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contents: `["hX1TEz_z877_XAtr3COaWg", "given_name", "Mario"]`

Claim `family_name`:

- SHA-256 Hash: `Q7TX7kL8CNUp3BFBKP5xxIuPu5gRgkO6HplM3E1iMIc`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contents: `["YWtI06xDdCyvTalcInTE3A", "family_name", "Rossi"]`

Claim `birth_date`:

- SHA-256 Hash: `oF2qeWAbKO_qWGQ5z-HGKeifl2PMIEMbJe8L-PJ-wko`
- Disclosure: `WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgImJpcnRoX2RhdGUiLCAiMTk4 MC0wMS0xM-CJd`
- Contents: `["-z34cJ1gC5UBPCIx8OhNiQ", "birth_date", "1980-01-10"]`

Claim `tax_id_code`:

- SHA-256 Hash: `Wq3gFfmC0I9Lefw1mh-Bk5XPRtoSCg9aE23uOhxakas`
- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgInRheF9pZF9jb2RlIiwgIlRJ TklULVhY-WFhYWFhYWFhYWFhYWFgiXQ`
- Contents: `["KNc5-Gk9CQh_TdGbqBKI7A", "tax_id_code", "TINIT-XXXXXXXXXXXXXXXX"]`

Claim `constant_attendance_allowance`:

- SHA-256 Hash: `JOQk0kuBSVk80rFlv9VGY-yiIzsfzEJKk3d4RROfzkM`
- Disclosure: `WyIyaFFtWXBIeVgtbVpKaHoyeHNVWWNRIiwgImNvbnN0YW50X2F0dGVuZGFu Y2VfY-Wxsb3dhbmNlIiwgdHJ1ZV0`
- Contents: `["2hQmYpHyX-mZJhz2xsUYcQ", "constant_attendance_allowance", true]`

The combined format for the (Q)EAA issuance is represented below:

```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2N2dM
N2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMiOiBbIjxJ
c3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19.eyJfc2QiOlsiRDRWa1dqbkEwV09ON0h
kQ0dGdFU4NjlNU3ZPUkhQZjhwNWZRUkQ1Z05qMCIsIkpPUWswa3VCU1ZrODByRmx2OVZ
```

HWS15aUl6c2Z6RUpLazNkNFJST2Z6a00iLCJRN1RYN2tMOENOVXAzQkZCS1A1eHhJdVB
1NWdSZ2tPNkhwbE0zRTFpTUljIiwiV3EzZ0ZmbUMwSTlMZWZ3MW1oLUJrNVhQUnRvU0N
nOWFFMjN1T2h4YWthcyIsIm9GMnFlV0FiS09fcVdHUTV6LUhHS2VpZmwyUE1JRU1iSmU
4TC1QSi13a28iLCJxYlJ0VUhwOU9heDlkbTVHZUtud19XMTJZdTFFMkRvVTZ3ckZQZWU
3YUJvIl0sImV4cCI6MTg4MzAwMDAwMCwiaXNzIjoiaHR0cHM6Ly9pc3N1ZXIuZXhhbXB
sZS5vcmciLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25vV1hGWkFmSGt4WnNSR0M5WHMiLCJ
pYXQiOjE2ODMwMDAwMDAsImlzc3VpbmdfYXV0aG9yaXR5IjoiQ3JlZGVudGlhbCBJc3N
1ZXIgT3JnYW5pemF0aW9uIiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V
4cGlyeSI6IjIwMzMtMDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4Ijo
1Njc4LCJ1cmkiOiJodHRwczovL2lzc3Vlci5leGFtcGxlLm9yZy9zdGF0dXMifX0sInZ
jdCI6InVybjppdC13YWxsZXQ6ZGlzYWJpbGl0eWNhcmQ6MSIsInZjdCNpbnRlZ3JpdHk
iOiIyZTQwYmNkNjc5OTAwODA4NWZmYjFhMWYzNTE3ZWZlZTMzNTI5OGZkOTc2YjNlNjU
1YmZiM2Y0ZWFhMTFkMTcxIiwidmVyaWZpY2F0aW9uIjp7InRydXN0X2ZyYW1ld29yayI
6Iml0X3dhbGxldCIsImFzc3VyYW5jZV9sZXZlbCI6Imh0dHBzOi8vdHJ1c3QtYW5jaG9
yLmV4YW1wbGUuaXQvbG9hL2hpZ2gifSwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOns
iandrIjp7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoiVENBRVIxOVp2dTNPSEY
0ajRXNHZmU1ZvSElQMUlMaWxEbHM3dkNlR2VtYyIsInkiOiJaeGppV1diWk1RR0hWV0t
WUTRoYlNJaXJzVmZ1ZWNDRTZ0NGpUOUYySFpRIn19fQ.rU0-nlNtkaqNS1h8h23YMs1m
BUIgJQDY6Ouq__Ih0gBEqviTrtkHZu_MgfLWRbAwnu59ubHo0PKkuy3c6D19SA~WyJrZ
2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImRvY3VtZW50X251bWJlciIsICJYWFhYWFhYW
FhYIl0~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImdpdmVuX25hbWUiLCAiTWFya
W8iXQ~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImZhbWlseV9uYW1lIiwgIlJvc3
NpIl0~WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgImJpcnRoX2RhdGUiLCAiMTk4MC
0wMS0xMCJd~WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgInRheF9pZF9jb2RlIiwgI
lRJTklULVhYWFhYWFhYWFhYWFgiXQ~WyIyaFFtWXBIeVgtbVpKaHoyeHNVWWNRIi
wgImNvbnN0YW50X2F0dGVuZGFuY2VfYWxsb3dhbmNlIiwgdHJ1ZV0~

Digital Credential Type Metadata Document

When provided, the Type Metadata Document MUST be a *JSON object* compliant with Section 6.2 of [SD-JWT-VC].

In addition to the properties defined in section 9.2 of [SD-JWT-VC] the following property SHOULD be included:

- `name`: A human-readable label for the claim, intended for end users.

The Credential Type Metadata JSON Document MAY be retrieved through a *well-known* endpoint. See Section 6.3.3 of SD-JWT-VC. This endpoint, provided by the Credential Issuer, MUST have the following format: `https://{Credential Issuer Domain}/.well-known/type-metadata`. The `vct` query parameter MUST be added to that endpoint. The Endpoint returns a `200 OK` status code and supports `application/json` as content type.

Below a non-normative example is given.

```
GET /.well-known/type-metadata?vct=urn%3Aeudi%3Apid%3Ait%3A1 HTTP/1.1
Host: pidprovider.example.it
```

```
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json

{
  "vct": "urn:eudi:pid:it:1",
  "name": "Person Identification Data",
  "description": "Digital version of Person Identification Data",
  ...
}
```

11.1.3 mdoc-CBOR Credential Format

When Digital Credentials are issued in mdoc-CBOR format, they MUST be based on the ISO/IEC 18013-5 standard.

The mdoc data elements MUST be encoded in CBOR as defined in **RFC 8949**.

This data model structures mdoc Digital Credentials into distinct components: namespaces (**nameSpaces**), and cryptographic proof (**issuerAuth**). Namespaces categorize and structure data elements (or attributes, see *Attribute Namespaces* (page 370)). While the cryptographic proof ensures integrity and authenticity through the Mobile Security Object (MSO).

The MSO securely stores cryptographic digests of attributes within the *nameSpaces*. This allows Relying Parties to validate disclosed attributes against corresponding **digestID** values without revealing the entire Credential. See *Mobile Security Object* (page 367) for details.

An mdoc-CBOR Digital Credential MUST be compliant with the following structure:

Parameter	Description	Reference
nameSpaces	*(map)*. The namespaces within which the data elements are defined. A Digital Credential MAY include multiple namespaces.	[ISO 18013-5#8.3.2.1.2]
issuerAuth	*(COSE_Sign1)*. Contains *Mobile Security Object* (MSO), a COSE Sign1 Document, issued by the Credential Issuer.	[ISO 18013-5#9.1.2.4]

Note

Mandatory mDL attributes utilize the standard namespace *org.iso.18013.5.1*. However, it MAY have a domestic namespace, such as *org.iso.18013.5.1.IT*, to include additional attributes defined in this implementation profile. Each namespace within the *nameSpaces*

MUST share the same issued document type (*docType*) value, which identifies the nature of the Digital Credential, as defined in the *issuerAuth*.

The structure of an mdoc-CBOR Credential is further elaborated in the following sections.

Mobile Security Object

The **issuerAuth** represents the *Mobile Security Object* which is a *COSE Sign1 Document* defined in **RFC 9052**. It has the following data structure:

- protected header
- unprotected header
- payload
- signature.

The **protected header** MUST contain the following parameter encoded in CBOR format:

Element	Description	Reference
1	*(int)*. Algorithm used to verify the cryptographic signature of the mdoc Digital Credential.	**RFC 9053**

Note

Only the signature algorithm MUST be present in the protected header, other elements SHOULD NOT be present in the protected header.

The **unprotected header** MUST contain the following parameters, unless otherwise specified:

Element	Description	Reference
4	*(tstr, OPTIONAL)*. Unique identifier of the Issuer JWK. Required when the Issuer of mdoc uses OpenID Federation.	*The Infrastructure of Trust* (page 109)
33	*(array)*. X.509 certificate chain about the Issuer. Required for X.509 certificate-based authentication.	**RFC 9360**

Note

The *x5chain* is included in the unprotected header with the aim to allow the Holder to update the X.509 certificate chain, related to the *Mobile Security Object* issuer, without invalidating the signature.

The **payload** MUST contain the *MobileSecurityObject*, without the *content-type* COSE Sign header parameter and encoded as a *byte string* (bstr) using the *CBOR Tag* 24.

The *MobileSecurityObject* MUST have the following attributes, unless otherwise specified:

Element	Description	Reference
docType	*(tstr)*. Format-encoded data identifier *credential_type_identifier* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). • When defined by an ISO standard, It MUST be a string of the form `iso.org.{iso-number}.{part}.{version}.{credential_type}` (e.g. for an mDL, the value MUST be `org.iso.18013.5.1.mDL`). • When defined at the european level, it MUST be a string of the form `eu.europa.ec.eudi.{credential_type}.{version}` (e.g., `eu.europa.ec.eudi.pid.1`). • When defined at national level, it MUST be a string of the form `{Trust Anchor reverse domain}.{credential_type}.{version}` (e.g., `it.wallet.trust-registry.pid.1`).	[ISO 18013-5#9.1.2.4]
version	*(tstr)*. Version of the *MobileSecurityObject*.	[ISO 18013-5#9.1.2.4]

continues on next page

Table 11.7 – continued from previous page

Element	Description	Reference
validityInfo	*(map, REQUIRED)*. Contains the *MobileSecurityObject* issuance and expiration datetimes. It includes the following sub-values: • **signed** *(tdate, OPTIONAL)*. The timestamp indicating when the *MobileSecurityObject* was signed. • **validFrom** *(tdate, OPTIONAL)*. Timestamp before which the *MobileSecurityObject* is not considered valid. When present, it MUST be equal to or later than the *signed* time. • **validUntil** *(tdate, REQUIRED)*. Timestamp after which the *MobileSecurityObject* is no longer considered valid. In accordance with [EIDAS-ARF] HLR **ISSU_12c** and **ISSU_12d** it MUST NOT be later than the expiration date of the Wallet Unit Attestation presented as part of the Digital Credential issuance process.	[ISO 18013-5#9.1.2.4]
digestAlgorithm	*(tstr)*. Identifier of the digest algorithm, which MUST match the algorithm defined in the protected header.	[ISO 18013-5#9.1.2.4]
valueDigests	*(map)*. Maps each namespace identifier to a set of digests, where each digest is keyed by a unique *digestID* and holds the digest value.	[ISO 18013-5#9.1.2.4]
deviceKeyInfo	*(map)*. Contains metadata about the Wallet Instance's public key. It MUST include the following sub-fields, unless otherwise specified: • **deviceKey** *(COSE_Key)*. Contains the public key parameters. • **keyAuthorizations** *(map, OPTIONAL)*. Defines authorizations for either full namespaces or individual data elements. • **keyInfo** *(map, OPTIONAL)*. Contains additional metadata about the key.	[ISO 18013-5#9.1.2.4]
status	*(map, OPTIONAL)*. REQUIRED only if the Digital Credential is long-lived. Format-encoded data identifier *location_status* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). Contains the MSO revocation information. If present, it includes a *status_list* based on the TOKEN-STATUS-LIST mechanism as defined in Section 6.3 of TOKEN-STATUS-LIST.	[ISO 18013-5#9.1.2.6]

Note

The private key related to the public key stored in the *deviceKey* map is used to sign the *DeviceSignedItems* and to prove the possession of the Digital Credential during the presentation phase (see the presentation phase with mdoc-CBOR).

Attribute Namespaces

The **nameSpaces** contains one or more *nameSpace* entries, each identified by a name. Within each **nameSpace**, it includes one or more *IssuerSignedItemBytes*, each encoded as a CBOR byte string with Tag 24 (#6.24(bstr .cbor)), which appears as 24(«... ») in diagnostic notation. It represents the disclosure information for each digest within the *Mobile Security Object* and MUST contain the following attributes:

Name	Description	Reference
digestID	*(uint)*. Reference value to one of the `ValueDigests` provided in the *Mobile Security Object*.	[ISO 18013-5#9.1.2.5]
random	*(bstr)*. Random byte value used as salt for the hash function. This value MUST be different for each *IssuerSignedItem* and it MUST have a minimum length of 16 bytes.	[ISO 18013-5#9.1.2.5]
elementIdentifier	*(tstr)*. Data element identifier.	[ISO 18013-5#8.3.2.1.2.3]
elementValue	*(any)*. Data element value.	[ISO 18013-5#8.3.2.1.2.3]

Digital Credential mdoc-CBOR Metadata Attributes

The following **elementIdentifiers** representing format-encoded metadata attributes are defined for Digital Credentials in mdoc-CBOR format within the respective *nameSpace*:

Element Identifier	Description	Reference
issuing_country	*(tstr, REQUIRED)*. Format-encoded data identifier *issuing_country* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). Alpha-2 country code as defined in [ISO 3166-1].	[ISO 18013-5#7.2]
issuing_authority	*(tstr, REQUIRED)*. Format-encoded data identifier *issuing_authority* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). The value MUST only use Latin1b characters and MUST have a maximum length of 150 characters.	[ISO 18013-5#7.2]

continues on next page

Table 11.9 – continued from previous page

Element Identifier	Description	Reference
issuance_date	*(tdate or full-date, OPTIONAL).* Format-encoded data identifier *issuance_date* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). This attribute pertains to the administrative issuance date, which is typically different from the technical issuance date expressed by the *MobileSecurityObject* parameters `signed` or `validFrom`.	Section 2.6 of the ARF PID Rulebook v1.3 [EIDAS-ARF].
expiry_date	*(tdate or full-date, OPTIONAL).* Format-encoded data identifier *expiry_date* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). It MUST be according to ISO 8601-1 YYYY-MM-DD format.	Section 3 of the ARF PID Rulebook v1.3 [EIDAS-ARF]
sub	*(uuid, OPTIONAL).* Identifies the subject of the mdoc Digital Credential (the User). The identifier MUST be opaque, MUST NOT correspond to any anagraphic data, and MUST NOT be derived from the User's anagraphic data through pseudonymization. Additionally, different Credentials issued to the same User or to different Users MUST NOT use the same *sub* value.	Domestic extension.
verification	*(map, OPTIONAL).* Format-encoded data identifier *verification* as defined in Section *Format-Agnostic Credential Metadata Attributes* (page 357). The CBOR map includes the following members: • `trust_framework` *(tstr, REQUIRED)*: trust framework used for User authentication. • `assurance_level` *(tstr, REQUIRED)*: level of identity assurance guaranteed during User authentication. The value MUST match with one of the values mapped in the `acr_values_supported` array of the credential-issuer-metadata:Credential Issuer Metadata.	Domestic extension.

Note

Digital Credential User-specific attributes are defined in the Catalog of Digital Credentials. User-specific attributes for mdoc Digital Credentials such as those used in mDL or PID are also included by referencing the appropriate *elementIdentifiers* defined in ISO/IEC 18013-5 or the EIDAS-ARF specification.

Note

Regardless of the Digital Credential type, the *sub* value MUST NOT be shown to the User, as it is not a User attribute. It is used for identification purposes by the Credential Issuers.

mdoc-CBOR Examples

A non-normative example of an mDL encoded in CBOR is shown below in binary encoding.

a26a6e616d65537061636573a2716f72672e69736f2e31383031332e352e318cd818a100a468646967657374494400
6672616e646f6d5820790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a9640671656c656d65
6e744964656e7469666965726b66616d696c795f6e616d656c656c656d656e7456616c756565526f737369d818a101
a468646967657374494404166726 16e646f6d58201442881e24514517333019ec24aecaa70bba927d7f2d38ad7cdc3c
e82d8561db71656c656d656e744964656e7469666965726a676976656e5f6e616d656c656c656d656e7456616c7565
654d6172696fd818a102a468646967657374494402667 2616e646f6d582051b4f3831d910861e81da746b221fd8949
8507476418cedc3709b5d28a7c41d071656c656d656e744964656e7469666965726a62697274685f646174656c656c
656d656e7456616c7565d903ec6a313938302d30312d3130d818a103a468646967657374494403667 2616e646f6d58
200c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f71656c656d656e744964656e7469
666965726a69737375655f646174656c656c656d656e7456616c7565d903ec6a323032352d30332d3237d818a104a4
6864696765737449440466726 16e646f6d58200a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e333967
0203178e71656c656d656e744964656e7469666965726b6578706972795f646174656c656c656d656e7456616c7565
d903ec6a323033302d30332d3237d818a105a468646967657374494405667 2616e646f6d58204b315ff17cf3a4754a
94a6cf1e9ddfdd99f6e86b177b74f173348968ca74e80b71656c656d656e744964656e7469666965726f6973737569
6e675f636f756e7472796c656c656d656e7456616c7565624954d818a106a468646967657374494406667 2616e646f
6d5820f8c82c4103f603435d0bc7762074ccc7c2c74925314a1fb5a8ab9cf2a960221f71656c656d656e744964656e
7469666965727169737375696e675f617574686f726974796c656c656d656e7456616c7565782849737469747546f
20506f6c696772616669636f2065205a656363612064656c6c6f20537461746fd818a107a468646967657374494407
6672616e646f6d58201f7a77a353da7bfc4da12691185249c31d421afd59ddac34f9e4fb4d92b8ec5071656c656d65
6e744964656e7469666965726b62697274685f706c6163656c656c656d656e7456616c756564526f6d61d818a108a4
6864696765737449440866726 16e646f6d582088e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6
c97cf21471656c656d656e744964656e7469666965726f646f63756d656e745f6e756d6265726c656c656d656e7456
616c7565695831323334353637d818a109a468646967657374494409667 2616e646f6d5820944758b43602b01ad6
8911b062349845492c04c6a78129bcf8cb5fb1396af2fc71656c656d656e744964656e7469666965726 8706f727472
6169746c656c656d656e7456616c7565590412ffd8ffe000104a46494600010101009000900000ffdb004300130d0e
110e0c13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241435f8260687175
7b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f7676767676767676767676767676
7676767676767676767676767676767676767676767676767676767676767676767676767676767676ffc0001108001800640301
2200021101031101ffc4001b0000030100030100000000000000000000000005060401020307ffc40032100001030303
0205020309000000000000010203040005110612211331141551617122410781a1163542527391b2c1f1ffc4001501
0101000000000000000000000000000000001ffc4001a1101010100030100000000000000000000000014111213161ff
da000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e7709723a1d0dae146ddfb
b3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb95ff00eb9ff14d215de66af089ce44b7
dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65d6b95a147381ea0d495b933275fe6bba75c114104a8ba4
10413e983dff004f5af5d34b4b4cde632d0bf1fd1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e85600
1ebc03c7ce29dd9eef1ef10fc447dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e545287
50946ec6704850cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493f602a075
559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7290875db4be88ab384bbbb
bfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30ab420a7a6d3240c718cf031fa9ef4c9ad550
205aa02951df4a1d6c8421b015b769db8c9229837ea2be8b1b0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2
edbd4584e7af9c64e3f96b9beb28d4ac40931e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4
a179c1c1e460938f9149ef655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dc
e15eed9b71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c5256a79e706d52
d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344e847dcac9dcb5abba8d145061d
43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a15f71dd27d451453e09d1a21450961cbb3ea8a956
433b781f1ce33dfed54f0e2b50a2b71d84ed6db18028a28175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a
66b726e33de9ea6f1b52b181c760724e47b514520a5a28a283ffd9d818a10aa468646967657374494 40a6672616e64
6f6d5820577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f71656c656d656e74496465
6e74696469657 27 6547269746965675f7 726976696c656765736c656c656d656e7456616c756581a375766568695

6c655f63617465676f72795f636f646561416a69737375655f64617465d903ec4b6a323032302d30392d31376b6578
706972795f64617465d903ec4b6a323033312d30362d3130d818a10ba468646967657374494 40b6672616e646f6d58
20fa21d3d890af5f4ea2760d08fd9a6256004cd5aa9d5e697ba5873fb0cddd555e71656c656d656e744964656e7469
6669657276756e5f646973374696e675f7369676e696e675f7369676e6c656c656d656e7456616c756561497467 2672e

The Diagnostic Notation of the CBOR-encoded mDL is given below.

```
{
"nameSpaces": {
  "org.iso.18013.5.1": [
    24(<<
        {
        "digestID": 0,
        "random": h'790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a96406',
        "elementIdentifier": "family_name",
        "elementValue": "Rossi"
        }
      >>),
    24(<<
        {
        "digestID": 1,
        "random": h'1442881e24514517333019ec24aecaa70bba927d7f2d38ad7cdc3ce82d8561db',
        "elementIdentifier": "given_name",
        "elementValue": "Mario"
        }
      >>),
    24(<<
        {
        "digestID": 2,
        "random": h'51b4f3831d910861e81da746b221fd89498507476418cedc3709b5d28a7c41d0',
        "elementIdentifier": "birth_date",
        "elementValue": 1004(1980-01-10)
        }
      >>),
    24(<<
        {
        "digestID": 3,
        "random": h'0c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f',
        "elementIdentifier": "issue_date",
        "elementValue": 1004(2025-03-27)
        }
      >>),
    24(<<
        {
        "digestID": 4,
        "random": h'0a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e3339670203178e',
        "elementIdentifier": "expiry_date",
        "elementValue": 1004(2030-03-27)
        }
      >>),
    24(<<
        {
```

```
    "digestID": 5,
    "random": h'4b315ff17cf3a4754a94a6cf1e9ddfdd99f6e86b177b74f173348968ca74e80b',
    "elementIdentifier": "issuing_country",
    "elementValue": "IT"
    }
  >>),
24(<<
    {
    "digestID": 6,
    "random": h'f8c82c4103f603435d0bc7762074ccc7c2c74925314a1fb5a8ab9cf2a960221f',
    "elementIdentifier": "issuing_authority",
    "elementValue": "Istituto Poligrafico e Zecca dello Stato"
    }
  >>),
24(<<
    {
    "digestID": 7,
    "random": h'1f7a77a353da7bfc4da12691185249c31d421afd59ddac34f9e4fb4d92b8ec50',
    "elementIdentifier": "birth_place",
    "elementValue": "Roma"
    }
  >>),
24(<<
    {
    "digestID": 8,
    "random": h'88e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6c97cf214',
    "elementIdentifier": "document_number",
    "elementValue": "XX1234567"
    }
  >>),
24(<<
    {
    "digestID": 9,
    "random": h'944758b43602b01ad68911b062349845492c04c6a78129bcf8cb5fb1396af2fc',
    "elementIdentifier": "portrait",
    "elementValue": h'ffd8ffe000104a464946000101010090009000000ffdb004300130d0e110e0c
    13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241435f82
    606871757b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f767676
    76767676767676767676767676767676767676767676767676767676767676767676767676767676
    76767676767676ffc000110800180064030122000211010311 01ffc4001b0000030100030100000000
    0000000000000000050604010203 07ffc400321000010303030205020309000000000000010203040
    0051106122113311415516171224107 81a1163542527391b2c1f1ffc400150101010000000000000
    0000000000000000001ffc4001a1101010100030100000000000000000000000014111213161ffda
    000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e7709723a1
    d0dae146ddfbb3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb95ff00
    eb9ff14d215de66af089ce44b7dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65d6b95a
    147381ea0d495b933275fe6bba75c114104a8ba410413e983dff004f5af5d34b4b4cde632d0bf1fd
```

```
      1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e856001ebc03c7ce29dd9eef1ef10fc4
      47dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e54528750946ec67048
      50cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493f602a0
      75559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7290875
      db4be88ab384bbbbbfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30ab420a
      7a6d3240c718cf031fa9ef4c9ad550205aa02951df4a1d6c8421b015b769db8c9229837ea2be8b1b
      0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2edbd4584e7af9c64e3f96b9beb28d4ac4093
      1e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4a179c1c1e460938f9149ef
      655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dce15eed9b
      71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c5256a79e
      706d52d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344e847dc
      ac9dcb5abba8d145061d43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a15f71dd
      27d451453e09d1a21450961cbb3ea8a956433b781f1ce33dfed54f0e2b50a2b71d84ed6db18028a2
      8175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a66b726e33de9ea6f1b52b181c760724e
      47b514520a5a28a283ffd9'
      }
    >>),
  24(<<
      {
      "digestID": 10,
      "random": h'577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f',
      "elementIdentifier": "driving_privileges",
      "elementValue": [
        {
          "vehicle_category_code": "A",
          "issue_date": 1004("2020-09-17"),
          "expiry_date": 1004("2031-06-10")
          }
        ]
      }
    >>),
  24(<<
      {
      "digestID": 11,
      "random": h'fa21d3d890af5f4ea2760d08fd9a6256004cd5aa9d5e697ba5873fb0cddd555e',
      "elementIdentifier": "un_distinguishing_sign",
      "elementValue": "I"
      }
    >>)
  ],
"org.iso.18013.5.1.IT": [
  24(<<
      {
      "digestID": 12,
      "random": h'0c3fe75be952ec3c2257031a71f2f54aeabfe7445705cec147fbb2c0f69ad561',
      "elementIdentifier": "sub",
      "elementValue": "3B4hK2m7fA9TdVzqLrGp6W8XyJ1sNtQc"
```

```
        }
      >>),
    24(<<
        {
        "digestID": 13,
        "random": h'd22c6db3dd27e066deb2ace6161e47fc6abc7a87c84a10320f14bc66d6e08d49',
        "elementIdentifier": "verification",
        "elementValue": {
          "trust_framework": "it_wallet",
          "assurance_level": "https://trust-anchor.example.it/loa/high",
          }
        }
      >>)
    ]
  },
"issuerAuth": [
    << {1: -7} >>,
    {
        33: h'30820208308201afa00302010202142eb39c647c81836bcf79fa9cd0b201ec0bf52307300a0
        6082a8648ce3d0403023064310b3009060355040613025553311330110603550408 0c0a43616c6966
        6f726e6961311630140603550407 0c0d53616e204672616e636973636f31133011060355040a0c0a4
        d7920436f6d70616e7931133011060355040 30c0a6d79736974652e636f6d301e170d323530333237
        3135353532305a170d3235303430363135353532305a3064310b3009060355040613025553311330 1
        106035504080c0a43616c69666f726e6961311630140603550407 0c0d53616e204672616e63697363
        6f31133011060355040a0c0a4d7920436f6d70616e7931133011060355040 30c0a6d79736974652e6
        36f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004f33da72d0dd0009b62221b
        0e839099b12dab5e01021124ebf9060422e648f3c3ec6614a86da1e91e552b2ae35e04d3058ae82b5
        c65a7f1f26800cb4499652a09a33f303d303b0603551d1104343032863068747470733a2f2f637265
        64656e7469616c2d6973737565722e6f6964632d666564657261746 96f6e2e6f6e6c696e65300a060
        82a8648ce3d0403020347003044022 04d1f0819971652b79ebe4825547de3d5554d2f41410225e6b1
        3dab949cda125e022079ba71b823619e49719dce5daa565bf745d3d97e2b87c7f7d6a626f981e653ed'
    },
    << 24(<<
        {
            "version": "1.0",
            "digestAlgorithm": "SHA-256",
            "valueDigests": {
              "org.iso.18013.5.1": {
                  0: h'f46b65d5060ad060ab9be62ff22ea8633437619ebdc7fa81f2d151159e92bffe',
                  1: h'e506545f6a6fd5d982670b4d62fc2b0688dc8f26754e7b0c574d63f5d72a85ac',
                  2: h'cfcf96fa12d100eeed5f00183d3b6a0888baa47eae85b5b95037eca7bbc0d07e',
                  3: h'8b0772252b0e06b611676b6b3402eb33bf866eb145e49f4d5f23215e6a047772',
                  4: h'14135c96693e2ab08d956876ee491357d906a6dd125557196dfb9811ba54aa8d',
                  5: h'86dcbd99233fbb84a9a2dce3a864a425e6e809300067a4475e3ea2a4d233dc74',
                  6: h'2e9512d35ea225e69e7b2180ecc1678dcc3e77a16e36427e64b4f0e2861b4d3a',
                  7: h'4efe55c36f6249d23c473a125afc5181aa3063393649478155497 1b72ff13700',
                  8: h'cc44a4f9983c5b0b1efc0e82e2867c8d5bbdf89c34bff16a1953c923bb4e4b3e',
```

```
                9: h'775eb2af0aa55f2071d62662b35c99698ae3bc0e2c4af5724ff88476cddd152f',
               10: h'915d0ad53dd23dace34968c263d307c04701a9bb9dc9865af91dc409786fd833',
               11: h'47d89ff4fb513044e6f2394236755ac0abf3e4f4a46f40454a458a59f8b7a6fb'
              },
            "org.iso.18013.5.1.IT": {
               12: h'16d2098702e896b4614dff1859bd3b42105cac2e62ce7f87dcacc249a656db32',
               13: h'755fd7c0f9272a8589c4a661a8aa80dc916018e500884eba316899d653fcb8d1'
              }
            },
          "deviceKeyInfo": {
            "deviceKey": {
               1: 2,
              -1: 1,
              -2: h'f96b29873b61f05403e2963a7ecbc799c9aab28d8a6629e5848cfdef85442866',
              -3: h'a9fef033a900c63e3894d8deb805a2a1fb55ef0d2b88e3c0d3336408186485ef'
               }
            },
          "docType": "org.iso.18013.5.1.mDL",
          "validityInfo": {
              "signed": 0("2025-03-27T00:00:00Z"),
              "validFrom": 0("2025-03-27T00:00:00Z"),
              "validUntil": 0("2026-03-27T00:00:00Z")
           },
         "status": {
          "status_list": {
            "idx": 1340,
            "uri": "https://statusprovider.example.org/statuslists/1"
          }
       }
       }
   >>) >>,
   h'd09f9acdf7a6be5e4aeb405bfb3b297b1b8003bcf52558a2f39fc6e5cffed40f18f49d2cc0e72a2a5645
     8d8aade591dee8d6540e639bca637f94bd9fa56f345c'
]
}
```

CBOR Acronyms

Acronym	Meaning
tstr	Text String
bstr	Byte String
int	Signed Integer
uint	Unsigned Integer
uuid	Universally Unique Identifier

continues on next page

Table 11.10 – continued from previous page

Acronym	Meaning
bool	Boolean (true/false)
tdate	Tagged Date (for example, Tag *0* is used to indicate a date/time string in **RFC 3339** format)

11.1.4 Cross-Format Credential Parameters Mapping

The following table provides a comparative mapping between the data structures of SD-JWT VC and mdoc-CBOR Digital Credentials. It outlines the key data elements and parameters used in each format, highlighting both commonalities and differences. In particular, it shows how core concepts - such as Credential Issuer information, validity, Cryptographic Binding, and disclosures - are represented in these Credential formats.

For SD-JWT-VC, parameters are marked with *(hdr)* if they are located in the JOSE header, and *(pld)* if they appear in the payload of the JWT. In mdoc-CBOR, these parameters are identified within the issuerAuth or nameSpaces structures.

Information Related To	SD-JWT-VC Parameters	mdoc-CBOR Parameters
Digital Credential type definition	vct (pld)	issuerAuth.doctype
Issuer		
	iss (pld)	-
	issuing_authority (pld)	nameSpaces.elementIdentifier.issuing_authority
	issuing_country (pld)	nameSpaces.elementIdentifier.issuing_country
Subject	sub (pld)	nameSpaces.elementIdentifier.sub
Validity period		
	iat (pld)	issuerAuth.validityInfo.signed
	exp (pld)	issuerAuth.validityInfo.validUntil
	nbf (pld)	issuerAuth.validityInfo.validFrom
	expiry_date (pld)	nameSpaces.elementIdentifier.expiry_date
Status mechanism		
	status_list (pld)	issuerAuth.status_list

continues on next page

Table 11.11 – continued from previous page

Information Related To	SD-JWT-VC Parameters	mdoc-CBOR Parameters
Signature		
	alg (hdr)	issuerAuth.1 (alg)
	kid (hdr)	issuerAuth.4 (kid)
Trust anchors		
	trust_chain (OID-FED) (hdr)	-
	x5c (hdr)	issuerAuth.33 (x5chain)
Cryptographic Binding	cnf.jwk (pld)	issuerAuth.deviceKeyInfo.deviceKey
Selective Disclosure	_sd_alg (pld)	issuerAuth.digestAlgorithm
	_sd (pld)	issuerAuth.valueDigests
Integrity		
	vct#integrity (pld)	
	Type_Metadata.extends#integrity (hdr)	-
Digital Credential format	typ (hdr)	
		-
Digital Credential auditability	verification (pld)	nameSpaces.elementIdentifier.verification
Disclosures		
	salt	
	claim name	nameSpaces
	claim value	

11.2 PID Data Model

The Person Identification Data (PID) is issued by the PID Provider according to national laws and it MUST be provided in both SD-JWT VC and mdoc-CBOR data format.

Note

The PID will be issued when the IT-Wallet will be fully operational and notified as EUDIW, the **IT-Wallet ID** is intended for access online services of national RPs only.

The main scope of the PID is allowing natural persons to be authenticated for access to a service or to a protected resource. The PID MUST be provided according to data model requirements defined in EU_2024/2977 and **Section 2 of the ARF PID Rulebook v1.3** [EIDAS-ARF], the User attributes provided within the Italian PID are the ones listed below:

- Current Family Name
- Current First Name
- Date of Birth
- Place of Birth
- Nationality
- Portrait
- Personal administrative number

In addition to the User attributes listed above, the PID includes also the following metadata attributes (EU_2024/2977 and **Section 2 of the ARF PID Rulebook v1.3** [EIDAS-ARF]):

- Issuing authority
- Issuing country
- Expiry Date
- Validity status information
- Identity and data proofing information

Some attributes, such as the *identity and data proofing information*, are provided as **domestic extensions** defined by the Italian IT-Wallet specification. It is NOT part of the ARF PID Rulebook (Annex 3.01, PID Rulebook v1.3), but is **permitted under ARF requirement PID_06**, which allows Member States to define additional domestic attributes beyond those specified in Commission Implementing Regulation (CIR) 2024/2977 (EU_2024/2977). In particular, the identity proofing information is REQUIRED for Italian PIDs to ensure:

- The evaluation of User authentication method used.
- The level of Assurance compliance of identity proofing during the enrollment process, according to the LoA defined by the eIDAS Regulation.
- The auditability upon the User attributes verification processes.

Attributes that are **domestic extensions** MUST be included in the **domestic namespaces** that are defined in Section *PID Data Model in SD-JWT VC Format* (page 382) and Section

PID Data Model in mdoc-CBOR Format (page 387) for SD-JWT VC and mdoc-CBOR PIDs respectively.

11.2.1 PID Data Model in SD-JWT VC Format

The SD-JWT VC PID defined in this specification MUST use the `vct` claim value set with `urn:eudi:pid:it:1`, according to the domestic PID extensions defined in the ARF PID Rulebook v1.3 (see also ARF HLR **PID_14**, Section 4.2, extending the base type `urn:eudi:pid:`).

Note

Transitional Phase:

During the transitional phase before full EUDIW operability, national implementations MAY use the `vct` value `urn:it-wallet:pid:1`. Upon reaching full EUDIW operability, all implementations MUST transition to the EUDI-compliant identifier `urn:eudi:pid:it:1` specified above.

According to EU_2024/2977 and **Section 4 of the ARF PID Rulebook v1.3** [EIDAS-ARF], the PID in SD-JWT VC format includes the following User Attributes:

Claim	Description	Reference
given_name	REQUIRED. *String*. Current First Name.	Section 5.1 of OIDC and Commission Implementing Regulation EU_2024/2977
family_name	REQUIRED. *String*. Current Family Name.	Section 5.1 of OIDC and Commission Implementing Regulation EU_2024/2977
birthdate	REQUIRED. *String*. Date of Birth. It MUST be set according to ISO8601-1 (YYYY-MM-DD format).	Commission Implementing Regulation EU_2024/2977
place_of_birth	REQUIRED. *JSON Object*. Place of Birth. At least one of *country*, *region*, *locality* MUST be present.	Commission Implementing Regulation EU_2024/2977

continues on next page

Table 11.12 – continued from previous page

Claim	Description	Reference
nationalities	REQUIRED. *Array of strings.* One or more alpha-2 country codes as specified in ISO 3166-1.	Commission Implementing Regulation EU_2024/2977
picture	REQUIRED. *String.* Facial image encoded as a data URL containing the base64-encoded JPEG portrait, compliant with the quality requirements for a full frontal image type as set out in ISO/IEC 39794-5 or, for backward compatibility, ISO/IEC 19794-5, clauses 8.2, 8.3 and 8.4. It MUST NOT include the headers or blocks specified in clause 5 of ISO/IEC 19794-5, except for the image data itself. Except where the User explicitly opts out, where applicable; in case of opt-out, the value MUST be empty, as specified in ARF HLR **PID_03**. Mandatory inclusion of the portrait attribute applies as of 24 months after entry into force of the Regulation amending EU_2024/2977.	Commission Implementing Regulation EU_2024/2977 and ARF PID Rulebook
personal_administrative_number	OPTIONAL. *String.* National unique identifier of a natural person. It MUST be set according to ETSI EN 319 412-1 Section 5.1.3. The only natural person semantic identifiers (**NAT-5.1.3-03**) supported whithin IT-Wallet are `PNO` and `TIN`. For IDANPR, `PNO` MUST be used, while for Fiscal Code, `TIN` MUST be used.	Commission Implementing Regulation EU_2024/2977

Note

Identity Matching and Identity Reconciliation

For the PID, `personal_administrative_number` is OPTIONAL. Relying Parties MUST NOT assume that a unique national identifier is always available. They MUST apply the identity matching and identity reconciliation rules defined in *Identity Matching and Identity Reconciliation* (page 538). When that identifier is presented, Unique National Identifier Binding applies (*CI_201* (page 776)). When it is not presented, Attribute-Based Binding on at least `family_name`, `given_name`, `birth_date` (or `birthdate`) and `place_of_birth` applies (*CI_202* (page 776)).

All the User attributed listed above MUST be selectively disclosable. In addition to the mandatory metadata attributes defined in *SD-JWT header JOSE Parameters Table* (page 359) and *SD-JWT Parameters Table* (page 359), the following metadata attributes are REQUIRED for a PID:

- **date_of_expiry**

- **sub** (domestic extension)
- **iat**
- **cnf**
- **status**
- **verification** (domestic extension)

SD-JWT-VC PID Non-Normative Example

In the following, the non-normative example of the payload of a PID represented in JSON format.


```
{
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "exp": 1883000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "given_name": "Mario",
  "family_name": "Rossi",
  "birthdate": "1980-01-10",
  "place_of_birth": {
    "locality": "Roma"
  },
  "nationalities": [
    "IT"
  ],
  "picture": "..."
}
```


The corresponding SD-JWT version for PID is given by

```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
  "_sd": [
    "GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY",
    "Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI",
    "MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk",
    "tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU",
    "uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8",
    "R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A"
  ],
  "exp": 1883000000,
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

The disclosure list is presented below.

Claim `given_name`:

- SHA-256 Hash: `Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contents: `["kghte5MDNHbQfdJHp88pCA", "given_name", "Mario"]`

Claim `family_name`:

- SHA-256 Hash: `MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contents: `["hX1TEz_z877_XAtr3COaWg", "family_name", "Rossi"]`

Claim `birthdate`:

- SHA-256 Hash: `uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0ZSIsICIxOTgw LTAxLTEwIl0`
- Contents: `["YWtI06xDdCyvTalcInTE3A", "birthdate", "1980-01-10"]`

Claim `place_of_birth`:

- SHA-256 Hash: `tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU`
- Disclosure: `WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBsYWNlX29mX2JpcnRoIiwg eyJsb2Nhb Gl0eSI6ICJSb21hIn1d`
- Contents: `["XcXlPVCqjHNveBCnlVPYgA", "place_of_birth", {"locality": "Roma"}]`

Claim `nationalities`:

- SHA-256 Hash: `GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY`
- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBb IklUIl1d`
- Contents: `["KNc5-Gk9CQh_TdGbqBKI7A", "nationalities", ["IT"]]`

Claim `picture`:

- SHA-256 Hash: `R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A`
- Disclosure: `WyJRaDhMbU40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFn ZS9qc GVnO2Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ`
- Contents: `["Qh8LmN4xR7vP2cKjT5sZYA", "picture", " 4AAQSkZJRgABAQAAAQABAAD..."]`

The combined format for the PID issuance is given by:

```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2
N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMi
OiBbIjxJc3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19eyJfc2QiOlsiR0hZanVH
VXRoanRCNHE0T3pfWlNHUG1Db2tMT3B2MmtwRk56ejFMZkZVWSIsIkprYmo4YUxy
LXoyX2MtSFZ4Q2JpdzZZWEZOSGl5TFN2MXhHak44bFJvZ0kiLCJNV0p1ZlF6X0RG
V2M5Y1I0eXhxOFhxbVRaZmdsa2cyRDJTeGEzVUZONFFrIiwidEk1czJBX0V6Nm9a
djZwbFp6VVBqWUFMLVNKR2lBVUZ5UmJoekxzbHVHVSIsInVJYXBVbERUS3NCNXdO
N0JGNnh1Qk5UdGw3NGdsNWlDdV9hUTVuajNZTDgiLCJSNng5bzBqNG0zTDRQcTZ4
WWRLM3JQNG5OcTh2SjJiN3NGcVQ5Y1V3STdBIl0sImV4cCI6MTg4MzAwMDAwMCwi
aXNzIjoiaHR0cHM6Ly9waWRwcm92aWRlci5leGFtcGxlLm9yZyIsInN1YiI6Ik56
YkxzWGg4dURDY2Q3bm9XWEZaQWZIa3hac1JHQzlYcyIsImlhdCI6MTY4MzAwMDAw
MCwiaXNzdWluZ19hdXRob3JpdHkiOiJQSUQgUHJvdmlkZXIgT3JnYW5pemF0aW9u
IiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V4cGlyeSI6IjIwMzMt
MDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4IjoxMjM0LCJ1cmki
OiJodHRwczovL3BpZHByb3ZpZGVyLmV4YW1wbGUub3JnL3N0YXR1cyJ9fSwidmN0
IjoidXJuOmV1ZGk6cGlkOml0OjEiLCJ2Y3QjaW50ZWdyaXR5IjoiNTVmNWZlNTdm
MWU0OTFkNDlmMzM2NzI3ODQ1Mjg1MzJkOWQiLCJ2ZXJpZmljYXRpb24iOnsidHJ1
c3RfZnJhbWV3b3JrIjoiaXRfY2llIiwiYXNzdXJhbmNlX2xldmVsIjoiaHR0cHM6
Ly90cnVzdC1hbmNob3IuZXhhbXBsZS5pdC9sb2EvaGlnaCJ9LCJfc2RfYWxnIjoi
c2hhLTI1NiIsImNuZiI6eyJqd2siOnsia3R5IjoiRUMiLCJjcnYiOiJQLTI1NiIs
IngiOiJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbERsczd2Q2VHZW1j
IiwieSI6Ilp4amlXV2JaTVFHSFZXS1ZRNGhiU0lpcnNWZnVlY0NFNnQ0alQ5RjJI
WlEifX19~WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCA
iTWFyaW8iXQ~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1l
IiwgIlJvc3NpIl0~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0
ZSIsICIxOTgwLTAxLTEwIl0~WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBs
YWNlX29mX2JpcnRoIiwgeyJsb2NhbGl0eSI6ICJSb21hIn1d~WyJLTmM1LUdrOUN
RaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIklUIl1d~WyJRaDhMbU
40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFnZS9qcGVnO2
Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ~
```


11.2.2 PID Data Model in mdoc-CBOR Format

The PID in mdoc-CBOR format MUST use the **docType** `eu.europa.ec.eudi.pid.1` in compliance with ARF HLR **PID_04**.

The PID attributes MUST be encoded as specified in **Section 3 of the ARF PID Rulebook v1.3** [EIDAS-ARF] and organized in the following namespaces:

- **Standard ARF PID attributes**: namespace `eu.europa.ec.eudi.pid.1`
- **Italian domestic extensions**: namespace `eu.europa.ec.eudi.pid.it.1`

According to EU_2024/2977 and **Section 3 of the ARF PID Rulebook v1.3** [EIDAS-ARF], the PID in mdoc-CBOR format includes the following User Attributes:

elementIdentifier	Description	Namespace
given_name	REQUIRED. *(tstr)*. Current First Name.	`eu.europa.ec.eudi.pid.1`
family_name	REQUIRED. *(tstr)*. Current Family Name.	`eu.europa.ec.eudi.pid.1`
birth_date	REQUIRED. *(full-date)*. Date of Birth. It MUST be encoded as full-date string according to **RFC 8949**.	`eu.europa.ec.eudi.pid.1`
place_of_birth	REQUIRED. *(map)*. Place of Birth. At least one of `country`, `region`, `locality` MUST be present.	`eu.europa.ec.eudi.pid.1`
nationality	REQUIRED. *(array of tstr)*. One or more Alpha-2 country codes as specified in ISO 3166-1. Encoded as CDDL type `nationalities` (array of country codes).	`eu.europa.ec.eudi.pid.1`
portrait	REQUIRED. *(bstr)*. Facial image in JPEG format, compliant with the quality requirements for a full frontal image type as set out in ISO/IEC 39794-5 or, for backward compatibility, ISO/IEC 19794-5, clauses 8.2, 8.3 and 8.4. It MUST NOT include the headers or blocks specified in clause 5 of ISO/IEC 19794-5, except for the image data itself. Except where the User explicitly opts out, where applicable; in case of opt-out, the value MUST be empty, as specified in ARF HLR **PID_03**. Mandatory inclusion of the portrait attribute applies as of 24 months after entry into force of the Regulation amending EU_2024/2977.	`eu.europa.ec.eudi.pid.1`
personal_administrative_number	OPTIONAL. *(tstr)*. National unique identifier of a natural person generated by ANPR.	`eu.europa.ec.eudi.pid.1`

In addition to the mandatory metadata attributes defined in *MobileSecurityObject Table* (page 368) and *mdoc-CBOR Metadata Attributes Table* (page 370), the following metadata attributes are REQUIRED for a PID:

Attribute	Location
expiry_date	`eu.europa.ec.eudi.pid.1` namespace
sub	`eu.europa.ec.eudi.pid.it.1` namespace
validityInfo.signed	MobileSecurityObject
verification	`eu.europa.ec.eudi.pid.it.1` namespace
status	MobileSecurityObject (as defined in Section 6.3 of TOKEN-STATUS-LIST)

Note

Key differences from SD-JWT encoding:

The ARF PID Rulebook v1.3 uses different claim names between SD-JWT and mdoc-CBOR formats:

- mdoc uses `birth_date` (not `birthdate` as in SD-JWT)
- mdoc uses `expiry_date` (not `date_of_expiry` as in SD-JWT)
- mdoc uses `nationality` (not `nationalities` as in SD-JWT). Note: both formats encode the value as an array of country codes.
- mdoc uses `portrait` (not `picture` as in SD-JWT). Note: SD-JWT encodes the value as a data URL; mdoc encodes the JPEG bytes as `bstr`.

See Section 3.1.1 (mdoc encoding) and Section 4.1.1 (SD-JWT encoding) of the ARF PID Rulebook v1.3 for the complete mapping.

mdoc-CBOR PID Non-Normative Example

A non-normative example of a PID in mdoc-CBOR format (diagnostic notation) is shown below:

```
{
   "nameSpaces": {
       "eu.europa.ec.eudi.pid.1": [
       24(<<
           {
           "digestID": 1,
           "random": h'E0B70BCEFBD43686F345C9ED429343AA',
           "elementIdentifier": "expiry_date",
           "elementValue": 1004("2024-02-22")
           }
       >>),
       24(<<
           {
           "digestID": 2,
           "random": h'960CB15A2EA9B68E5233CE902807AA95',
           "elementIdentifier": "issuing_country",
           "elementValue": "IT"
           }
       >>),
       24(<<
           {
           "digestID": 3,
           "random": h'9D3774BD5994CCFED248674B32A4F76A',
           "elementIdentifier": "issuing_authority",
           "elementValue": "Ministero dell'Interno"
           }
       >>),
```

```
24(<<
    {
    "digestID": 4,
    "random": h'EB12193DC66C6174530CDC29B274381F',
    "elementIdentifier": "given_name",
    "elementValue": "Mario"
    }
>>),
24(<<
    {
    "digestID": 5,
    "random": h'DB143143538F3C8D41DC024F9CB25C9D',
    "elementIdentifier": "family_name",
    "elementValue": "Rossi"
    }
>>),
24(<<
    {
    "digestID": 6,
    "random": h'6059FF1CE27B4997B4ADE1DE7B01DC60',
    "elementIdentifier": "birth_date",
    "elementValue": 1004("1956-01-12")% the tag 1004 defines the value
    is a full date
    }
>>),
24(<<
    {
    "digestID": 7,
    "random": h'FX8VVN1YN6NBH3CKYPJTNPJIVJ5KDMFA',
    "elementIdentifier": "place_of_birth",
    "elementValue": {"locality": "Roma"}
    }
>>),
24(<<
    {
    "digestID": 8,
    "random": h'RGLXZ5CHND8Y2UUMHXXMJWPCHNH9HVIX',
    "elementIdentifier": "nationality",
    "elementValue": ["IT"],
    }
>>),
24(<<
    {
    "digestID": 9,
    "random": h'71EAC48D2F9B635A0C17D4E82ABF05',
    "elementIdentifier": "portrait",
    "elementValue": h'FFD8FFE000104A4649460001010000010001000 0...'
```

```
        }
    >>)
    ],
    "eu.europa.ec.eudi.pid.it.1": [
        24(<<
            {
            "digestID": 10,
            "random": h'44DD0606C5F5FDD206H8C84H3F67F5HEEH',
            "elementIdentifier": "sub",
            "elementValue": "550e8400-e29b-41d4-a716-446655440000"
            }
        >>),
        24(<<
            {
            "digestID": 11,
            "random": h'33cc9495c4f4fcc195g7b73g2e56e4gddg',
            "elementIdentifier": "verification",
            "elementValue": {
                "trust_framework": "it_cie",
                "assurance_level": "https://trust-anchor.example.it/loa/high",
            }
            }
        >>)
    ]
},
"issuerAuth": [
      << {1: -7} >>, % protected header with the value alg:ES256
      {
          33: h'30820215308201BCA003020102021404AD30C…'% 33->X5chain:COSE X_509
      },
      <<
          24(<<
              {
              "docType": "eu.europa.ec.eudi.pid.1",
              "version": "1.0",
              "validityInfo": {
                  "signed": 0("2023-02-22T06:23:56Z"),
                  "validFrom": 0("2023-02-22T06:23:56Z"),
                  "validUntil": 0("2024-02-22T00:00:00Z")
              },
              "valueDigests": {
                  "eu.europa.ec.eudi.pid.1": {
                      1: h'0F1571A97FFB799CC8FCDF2BA4FC2909929…',
                      2: h'E2382149255AE8E955AF9B8984395…',
                      3: h'BBC77E6CCA981A3AD0C3E544EDF86…',
                      4: h'AA5D5C57B6E7D88FC6F98895E4F67…',
                      5: h'BB6E6C68D1B4B4EC5A2AE9206F5t4…',
```

```
                    6: h'F8A5966E6DAC9970E0334D8F75E25…',
                    7: h'1F1A0F7CCDD79b29EB4B22F96F54F…',
                    8: h'1A0F7EEDB46B37AS4N22R96D83F3F…',
                    9: h'71EAC48D2F9B635A0C17D4E82ABF5…'
                },
                "eu.europa.ec.eudi.pid.it.1": {
                    10: h'C8FF5E58G89FCE97G45533CE3E4B…',
                    11: h'F9EE4D36F67DBD75E23311AC1C29…',
                    12: h'A2BC5E47G78ECD86F34422BD2D3A…'
                }
            },
            "deviceKeyInfo": {
                "deviceKey": {
                    1: 2, % kty:EC2 (Eliptic curves with x and y coordinate pairs)
                    -1: 1, % crv:p256
                    -2: h'B820963964E53AF064686DD9218303494A…', % x-coordiantes
                    -3: h'0A6DA0AF437E2943F1836F31C678D89298E9…'% y-ccordiantes
                }
            },
            "status": {
                "status_list": {
                    "idx": 0,
                    "uri": "https://issuer.example.org/statuslists/1"
                }
            },
            "digestAlgorithm": "SHA-256"
            }
        >>)
    >>,
    h'1AD0D6A7313EFDC38FCD765852FA2BD43DEBF48BF5A580D'
    ]
}
```

11.3 IT-Wallet ID Data Model

The **IT-Wallet ID** (*Electronic Attestation of Person Identification Data (national scope)*) is an Electronic Attestation of Attributes (EAA) issued by the EAA Provider according to national laws. It is provided in SD-JWT VC format.

The IT-Wallet ID is intended **exclusively for national use** with Italian Relying Parties to access online services. It MUST NOT be used for cross-border interactions and does **not** constitute a PID under the European Digital Identity framework.

The main scope of the IT-Wallet ID is allowing natural persons to be authenticated for access to a service or to a protected resource within the national jurisdiction. The User attributes provided within the IT-Wallet ID are the ones listed below:

- Current Family Name
- Current First Name
- Date of Birth
- Place of Birth
- Nationality
- User identification number across public Relying Party services (for example the *taxpayer identification number*)

In addition to the User attributes listed above, the IT-Wallet ID includes also the following metadata attributes:

- Issuing authority
- Issuing country
- Expiry Date
- Validity status information
- Identity and data proofing information

The identity proofing information is REQUIRED for IT-Wallet ID to ensure:

- The evaluation of User authentication method used.
- The level of Assurance compliance of identity proofing during the enrollment process, according to the LoA defined by the eIDAS Regulation.
- The auditability upon the User attributes verification processes.

11.3.1 IT-Wallet ID Data Model in SD-JWT VC Format

The SD-JWT VC IT-Wallet ID defined in this specification MUST use the `vct` claim value set with `urn:it-wallet:eid:1`, where `eid` is the catalogue `credential_type` of IT-Wallet ID.

The IT-Wallet ID in SD-JWT VC format includes the following User Attributes:

Claim	Description	Reference
given_name	REQUIRED. *String*. Current First Name.	Section 5.1 of OIDC and Commission Implementing Regulation EU_2024/2977
family_name	REQUIRED. *String*. Current Family Name.	Section 5.1 of OIDC and Commission Implementing Regulation EU_2024/2977
birthdate	REQUIRED. *String*. Date of Birth. It MUST be set according to ISO8601-1 (YYYY-MM-DD format).	Commission Implementing Regulation EU_2024/2977
place_of_birth	REQUIRED. *JSON Object*. Place of Birth. At least one of *country*, *region*, *locality* MUST be present.	Commission Implementing Regulation EU_2024/2977
nationalities	REQUIRED. *Array of strings*. One or more alpha-2 country codes as specified in ISO 3166-1.	Commission Implementing Regulation EU_2024/2977
personal_administrative_number	REQUIRED if `tax_id_code` is not present, OPTIONAL otherwise. *String*. National unique identifier of a natural person generated by ANPR in string format.	Commission Implementing Regulation EU_2024/2977
tax_id_code	REQUIRED if `personal_administrative_number` is not present, OPTIONAL otherwise. *String*. National tax identification code of natural person as a String format. It MUST be set according to ETSI EN 319 412-1. For example `TINIT-<ItalianTaxIdentificationNumber>`.	Domestic extension

Note

Identity Matching

For the IT-Wallet ID, at least one of `tax_id_code` and `personal_administrative_number` is present. The Relying Party MUST apply Unique National Identifier Binding as specified in *Identity Matching and Identity Reconciliation* (page 538) (*CI_201* (page 776)).

All the User attributed listed above MUST be selectively disclosable. In addition to the mandatory metadata attributes defined in *SD-JWT header JOSE Parameters Table* (page 359) and *SD-JWT Parameters Table* (page 359), the following metadata attributes are REQUIRED for a IT-Wallet ID:

- **date_of_expiry**
- **sub** (domestic extension)
- **iat**
- **cnf**
- **status**
- **verification** (domestic extension)

SD-JWT-VC IT-Wallet ID Non-Normative Example

In the following, the non-normative example of the payload of a IT-Wallet ID represented in JSON format.


```
{
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "exp": 1883000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "given_name": "Mario",
  "family_name": "Rossi",
  "birthdate": "1980-01-10",
  "place_of_birth": {
    "locality": "Roma"
  },
```

```
  "nationalities": [
    "IT"
  ],
  "picture": "..."
}
```


The corresponding SD-JWT version for IT-Wallet ID is given by


```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
  "_sd": [
    "GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY",
    "Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI",
    "MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk",
    "tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU",
    "uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8",
    "R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A"
  ],
  "exp": 1883000000,
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
```

```
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

The disclosure list is presented below.

Claim `given_name`:

- SHA-256 Hash: `Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contents: `["kghte5MDNHbQfdJHp88pCA", "given_name", "Mario"]`

Claim `family_name`:

- SHA-256 Hash: `MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contents: `["hX1TEz_z877_XAtr3COaWg", "family_name", "Rossi"]`

Claim `birthdate`:

- SHA-256 Hash: `uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0ZSIsICIxOTgw LTAxLTEwIl0`
- Contents: `["YWtI06xDdCyvTalcInTE3A", "birthdate", "1980-01-10"]`

Claim `tax_id_code`:

- SHA-256 Hash: `_C7hoKFt0kV190v2GXIwLUIiDbc_7LcyofQmgDfute8`
- Disclosure: `WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgInRheF9pZF9jb2RlIiwgIlRJ TklULVhY-WFhYWFhYWFhYWFhYWFgiXQ`
- Contents: `["-z34cJ1gC5UBPCIx8OhNiQ", "tax_id_code", "TINIT-XXXXXXXXXXXXXXXX"]`

Claim `place_of_birth`:

- SHA-256 Hash: `tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU`
- Disclosure: `WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBsYWNlX29mX2JpcnRoIiwg eyJsb2Nhb Gl0eSI6ICJSb21hIn1d`
- Contents: `["XcXlPVCqjHNveBCnlVPYgA", "place_of_birth", {"locality": "Roma"}]`

Claim `nationalities`:

- SHA-256 Hash: `GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY`

- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBb IklUIl1d`
- Contents: `["KNc5-Gk9CQh_TdGbqBKI7A", "nationalities", ["IT"]]`

The combined format for the IT-Wallet ID issuance is given by:


```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2
N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMi
OiBbIjxJc3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19eyJfc2QiOlsiR0hZanVH
VXRoanRCNHE0T3pfWlNHUG1Db2tMT3B2MmtwRk56ejFMZkZVWSIsIkprYmo4YUxy
LXoyX2MtSFZ4Q2JpdzZZWEZOSGl5TFN2MXhHak44bFJvZ0kiLCJNV0p1ZlF6X0RG
V2M5Y1I0eXhxOFhxbVRaZmdsa2cyRDJTeGEzVUZONFFrIiwidEk1czJBX0V6Nm9a
djZwbFp6VVBqWUFMLVNKR2lBVUZ5UmJoekxzbHVHVSIsInVJYXBVbERUS3NCNXdO
N0JGNnh1Qk5UdGw3NGdsNWlDdV9hUTVuajNZTDgiLCJSNng5bzBqNG0zTDRQcTZ4
WWRLM3JQNG5OcTh2SjJiN3NGcVQ5Y1V3STdBIl0sImV4cCI6MTg4MzAwMDAwMCwi
aXNzIjoiaHR0cHM6Ly9waWRwcm92aWRlci5leGFtcGxlLm9yZyIsInN1YiI6Ik56
YkxzWGg4dURDY2Q3bm9XWEZaQWZIa3hac1JHQzlYcyIsImlhdCI6MTY4MzAwMDAw
MCwiaXNzdWluZ19hdXRob3JpdHkiOiJQSUQgUHJvdmlkZXIgT3JnYW5pemF0aW9u
IiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V4cGlyeSI6IjIwMzMt
MDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4IjoxMjM0LCJ1cmki
OiJodHRwczovL3BpZHByb3ZpZGVyLmV4YW1wbGUub3JnL3N0YXR1cyJ9fSwidmN0
IjoidXJuOmV1ZGk6cGlkOml0OjEiLCJ2Y3QjaW50ZWdyaXR5IjoiNTVmNWZlNTdm
MWU0OTFkNDlmMzM2NzI3ODQ1Mjg1MzJkOWQiLCJ2ZXJpZmljYXRpb24iOnsidHJ1
c3RfZnJhbWV3b3JrIjoiaXRfY2llIiwiYXNzdXJhbmNlX2xldmVsIjoiaHR0cHM6
Ly90cnVzdC1hbmNob3IuZXhhbXBsZS5pdC9sb2EvaGlnaCJ9LCJfc2RfYWxnIjoi
c2hhLTI1NiIsImNuZiI6eyJqd2siOnsia3R5IjoiRUMiLCJjcnYiOiJQLTI1NiIs
IngiOiJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbERsczd2Q2VHZW1j
IiwieSI6Ilp4amlXV2JaTVFHSFZXS1ZRNGhiU0lpcnNWZnVlY0NFNnQ0alQ5RjJI
WlEifX19~WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCA
iTWFyaW8iXQ~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1l
IiwgIlJvc3NpIl0~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0
ZSIsICIxOTgwLTAxLTEwIl0~WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBs
YWNlX29mX2JpcnRoIiwgeyJsb2NhbGl0eSI6ICJSb21hIn1d~WyJLTmM1LUdrOUN
RaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIklUIl1d~WyJRaDhMbU
40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFnZS9qcGVnO2
Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ~
```


11.4 Digital Credential Lifecycle

The Credential Issuer is responsible for creating and issuing Digital Credentials, as well as managing their lifecycle and validity status.

The Authentic Source is the entity responsible for the management and provisioning of User's attributes to Credential Issuers. There is a relationship between the lifecycle of the attributes managed by the Authentic Source and the Digital Credential lifecycle managed by the Credential Issuer. Indeed, one of the reasons for revocation or suspension of Digital Credentials is the

update/revocation or suspension of the attributes contained in the Digital Credential. In IT Wallet, the provisioning of User's attributes and the notification of updates or changes in the state of the attributes are exchanged using the PDND infrastructure (see relative sections for more details).

Fig. 11.1 shows the states and transitions for Digital Credentials. It includes four distinct states: **Issued**, **Valid**, **Expired**, and **Revoked**. While, in case of (Q)EAAs there is an additional state: **Suspended**. A Digital Credential in all states can be deleted (**PID/(Q)EAA DEL**) and this ends its lifecycle.

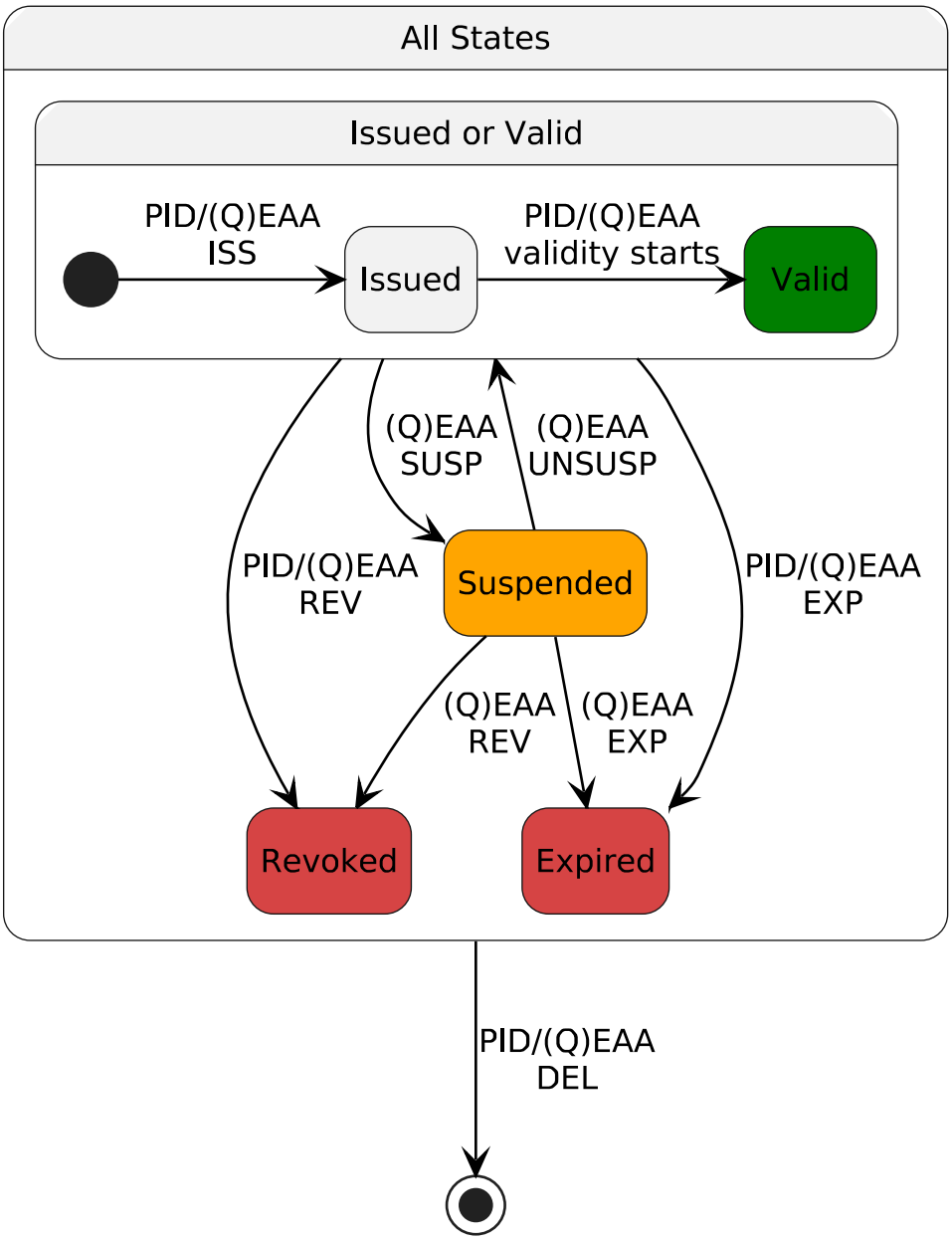


Fig. 11.1: Digital Credential State Transactions.

Note

Users MAY present a Digital Credential in any state, it is up to the Relying Party's policy to accept a not Valid Digital Credential. An example of this scenario is when a Relying Party needs to verify that the User is not a minor. In this case, even if the User presents an **Issued/Expired/Revoked** or **Suspended** Digital Credential, the age claim is still reliable.

Note

While **Issued**, **Valid**, **Expired**, **Revoked** are explicitly mentioned in the ARF (see Figure 5 of ARF v1.4), **Suspended** is implicitly present in EIDAS-ARF. This specification explicitly considers it.

11.4.1 Credential Transitions

Credential Transition to Issued

For the state machine to start, the Wallet Instance MUST be in either the **Operational** or **Valid** state, enabling Digital Credentials to be issued to it. The state machine begins with the **Issued** state, when an issuance process is triggered and, as a result, a Digital Credential is issued to the Wallet Instance (**PID/(Q)EAA ISS**). Please refer to *Digital Credential Issuance* (page 419).

Credential Transition to Valid

A Digital Credential changes to **Valid** state when:

- it reaches its start date of validity;
- an unsuspension process is triggered if the (Q)EAA has been suspended.

Credential Transition to Expired

A Digital Credential naturally transitions to the **Expired** state when it automatically expires upon reaching its end date of validity (**PID/(Q)EAA EXP**), indicating they are no longer valid for use.

If a Digital Credential is **Expired** the Wallet Instance SHOULD notify the User the Digital Credential has expired and the User MAY delete it (**PID/(Q)EAA DEL**). This ends its life-cycle.

Credential Transition to Revoked

A Digital Credential changes from **Issued**, **Valid** or **Suspended** states to **Revoked** state when it is actively revoked by the Credential Issuer by a revocation process (**PID/(Q)EAA REV**). The Relying Parties SHOULD no longer consider usable a particular Digital Credential when it is **Revoked**, even though it is still valid temporally and contains a valid Credential Issuer signature. Revocation can occur in the following cases:

- for technical security reasons relating to the compromise of cryptographic material;
- in case of explicit User requests;
- as a consequence of an attribute update by Authentic Sources;
- in case of a revocation of the attributes contained in the Digital Credential notified by the Authentic Source;
- death of the User;
- revocation of Wallet Instance to which the Digital Credential was issued;
- illegal activities of the User reported by Judicial or Supervisory Bodies.

In the case of PID and IT-Wallet ID only, the following cases are in addition to those listed above:

- detection of a breach of the digital identity issued by an Identity Provider and used to authenticate the User during the PID/IT-Wallet ID Issuance;
- as a result of obtaining a new PID/IT-Wallet ID on a new Wallet Instance from the same Wallet Provider that has provided the Wallet Instance containing a PID/IT-Wallet ID previously issued.

Note

A (Q)EAA Provider MAY revoke a (Q)EAA in case of PID/IT-Wallet ID revocation.

When a Digital Credential is **Revoked** it cannot transition back to **Valid**, the Wallet Instance SHOULD notify the User the Digital Credential has been revoked and the User MAY delete it (**PID/(Q)EAA DEL**). This ends its lifecycle.

Credential Transition to Suspended

A (Q)EAA changes from **Issued** or **Valid** states to **Suspended** state when it is suspended by the Credential Issuer (**(Q)EAA SUSP**). The (Q)EAA remains **Suspended** until it is restored to the **Issued** or **Valid** state (**(Q)EAA UNSUSP**) depending on the previous state, i.e. the conditions leading to its suspension are resolved, or it changes in **Revoked**, **Expired** or it is deleted. The suspension of a (Q)EAA MAY be:

- Use case driven, based on the validity status of the attributes contained in the (Q)EAA. In this case, an Authentic Source MUST notify the Credential Issuer of any changes in the state of the attributes attested by the (Q)EAA.
- Explicitly requested by the User.

11.4.2 Digital Credential Lifecycle in a Batch

For Digital Credentials issued in a single batch, each Credential immediately enters its own lifecycle state machine. All state transitions (Issued → Valid → Expired/Suspended/Revoked) still occur on a per Credential basis, using Credential's individual parameters (e.g., validity dates, Status List).

11.4.3 Credential Lifecycle Management

While Fig. 11.1 shows the different states a Digital Credential may acquire during its lifecycle, Fig. 11.2 shows the point of view of Wallet Instances and Credential Issuers in managing the Digital Credential lifecycle and the effect on their local storage.

A User, through the Wallet Instance, is able to acquire a new Digital Credential (**Credential Acquisition**) performing the **PID/(Q)EAA ISS** process. This MUST result in the storage of a Digital Credential in the **Issued/Valid** state, and delete it when it is not needed anymore or it is **Expired/Revoked** (**Credential Deletion**). Until the **Credential Deletion**, a Digital Credential can be presented to Relying Parties, this operation will not affect its lifecycle.

A Credential Issuer instead is responsible for:

- **Digital Credential Generation**: the Digital Credential is generated as a consequence of an issuance request and MUST be added to the local storage of the Credential Issuer after the successful issuance.
- **Digital Credential Revocation/Suspension/Unsuspension** (**PID/(Q)EAA REV** and **(Q)EAA SUSP/UNSUSP**): for technical security reasons or triggered by external entities (e.g., Users and Authentic Sources) the Digital Credential state MUST be locally updated.

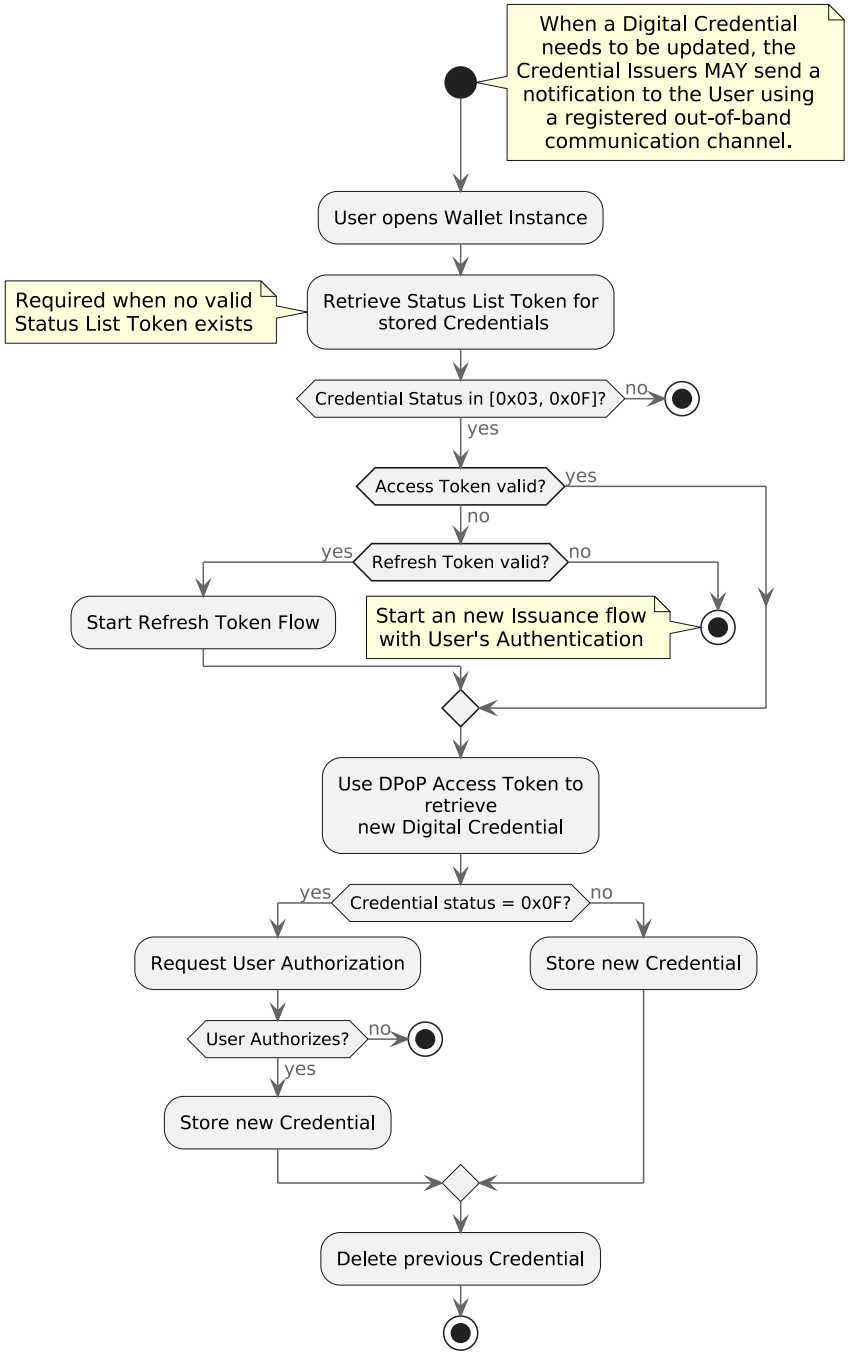


Fig. 11.2: Digital Credential Lifecycle Management.

- **Data Purging**: after reaching the **Expired** state, and based on the Credential Issuer retention policies, Digital Credentials MUST be removed from the local storage of the Credential Issuer.

Digital Credential Revocation and Suspension

This section describes the flows to request a Digital Credential status update (i.e. revocation or suspension), involved entities, and validation mechanisms for Digital Credentials in the IT-Wallet system.

As highlighted in Section *Digital Credential Lifecycle* (page 398), a Digital Credential's lifecycle is affected by:

- The lifecycle of its storing Wallet Instance
- The validity of Attributes managed by Authentic Sources
- For PIDs only, the status of the Digital Identity used for User authentication

External user-related factors can also influence a Digital Credential's lifecycle, such as:

- Explicit request from the Digital Credential holder
- User's death
- Illegal activities

Entities Involved

While the Credential Issuer MUST directly manage the validity status of Digital Credentials it has issued, other actors MAY trigger the Digital Credential revocation/suspension process:

- Users, through:
  - Their Wallet Instance
  - Web service provided by the Issuer
- The Authentic Source when Credential attributes are updated or change validity status
- The Wallet Provider when revoking a Wallet Instance
- The Identity Provider if the Digital Identity used for PID/IT-Wallet ID issuance is stolen or compromised
- Legal authorities or the Supervisory Body in cases of proven illegal activities

The following figure shows an entity relationships diagram relating to the Update Flow status.

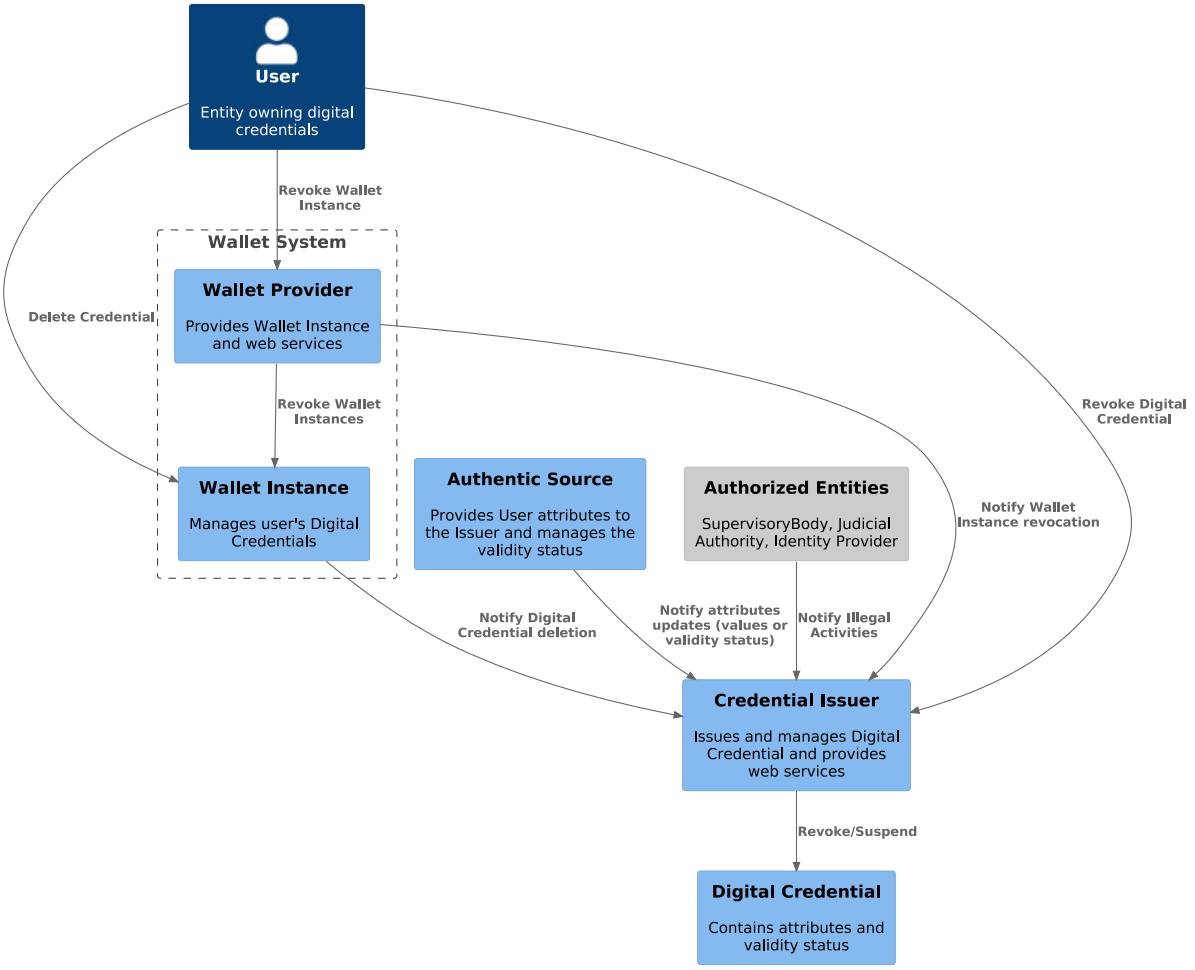


Fig. 11.3: Entities involved in Credential Revocation Flow.

Status Update Flows

This section describes the main flows for managing Digital Credential Status Updates by the Issuer, in particular Status Update:

- related to the User;
- triggered by a Wallet Instance;
- triggered by a Wallet Provider;
- triggered by an Authentic Source.

Note

Detailed Status Update Flows for Identity Providers, legal authorities, and the Supervisory Body will be covered in future versions of the technical specification.

Status Update related to the User

Users MAY change their Digital Credential validity status by:

1. Deleting the Digital Credential from their Wallet Instance: the Wallet Instance SHOULD prompt the User to be optionally notify the Credential Issuer about the User's intention to revoke the Digital Credential. When the User uses this functionality, the notification to be sent to the Credential Issuer MUST use the Notification Endpoint provided by the Issuer, as described in Section *Status Update by Wallet Instance* (page 407).
2. Using the Issuer's web portal:
   a. Users MAY access a secure area with at least the same Level of Assurance used during the issuance phase.
   b. The Issuer MUST allow Users to:
   - View all their Digital Credentials contained in the Issuer's database.
   - Verify data authenticity.
   - View and update validity status (revoke their Digital Credentials and, if it is supported by the Issuer, suspend them).

In addition, when Users detect incorrect data in an issued Digital Credential, the Wallet Instance SHOULD initiate a data correction request via the Notification Endpoint as specified in credential-issuance-endpoint:Data Correction using credential_failure. Upon confirmation of the discrepancy, the Issuer SHOULD follow the *Re-Issuance Flow* (page 448).

Note

If the User activates another Wallet Instance from the same Wallet Provider and using the same Wallet Solution and obtains a new PID/IT-Wallet ID, the previous PID/IT-Wallet ID MUST be revoked. In case of PID revocation the previous Wallet Instance MUST transition to operational status.

In case of the death of the User, Issuers MUST ensure that Digital Credentials and Wallet Instances owned by the User are revoked. The User's death triggers a change in the validity status of the User's identification attributes contained in the public registry (ANPR). The User's death MUST produce the PID/IT-Wallet ID revocation. Therefore, the Authentic Source of the PID/IT-Wallet ID (ANPR) MUST notify the PID/EAA Provider that the User's attributes are no longer valid due to the death of the User. The Authentic Source and the PID/EAA Provider MUST use the mechanisms provided in the Section *Status Update by Authentic Sources* (page 408).

Note

Future versions of this technical specification will define how the information to (Q)EAA Issuers are propagated, according to national regulation. Moreover, automated procedures for Credential revocation due to illegal activities will be defined in future specifications.

Status Update by Wallet Instance

When the User deletes a Digital Credential from the Wallet Instance, the Wallet Instance by default MUST NOT notify the Credential Issuer of this deletion event. Deleting a Digital Credential from the Wallet Instance only removes the local copy and does not change the validity status at the Issuer.

The Wallet Instance MAY inform the User, prior to deletion, that deletion is a local action and does not imply revocation at the Issuer, and MAY implement, under the User's explicit consent at deletion time, a notification feature to inform the Credential Issuer of the User's intention to revoke the Digital Credential.

If the User wants the Issuer to revoke a Digital Credential, the User SHOULD explicitly confirm this intention via the Wallet Instance's deletion prompt (when available), which MUST then notify the Credential Issuer; alternatively, the User MAY use the Issuer's web portal or other Issuer-provided channels.

When the revoked Credential is the PID/IT-Wallet ID, the Credential Issuer MUST send a notification of this event to the User within 24 hours. For any other Credential different from the PID/IT-Wallet ID, the Credential Issuer SHOULD send a notification of this event to the

User. The notification to the User MAY be implemented in several ways, such as using a User's email address, telephone number, or any other verified and secure communication channel. The notification to the User MUST also include all the information about the Credential revocation status. The method used for the notification to the User is out of scope of the current technical implementation profile. When the revocation occurs, the Credential Issuer MUST update the status of the Digital Credential accordingly. When the Notification Response sent by the Credential Issuer is successfully received by the Wallet Instance, the Wallet Instance MUST delete the Digital Credential.

Status Update by Wallet Providers

The Wallet Provider that, for any reason, revokes a Wallet Instance MUST ensure that the updated status is reflected in the status list of the related the Wallet Unit Attestation. In addition to what already defined in *Digital Credential Lifecycle* (page 398), the Credential Issuer MUST implement a monitoring mechanism of the current statuses of all the Wallet Unit Attestations related to the Wallet Instances to which the Credentials were issued. Afterwards the revocation of the Wallet Unit Attestation, the Credential Issuer proceeds with the revocation of all the Credentials issued to the Wallet Instance.

Status Update by Authentic Sources

Authentic Sources manage attributes separately from Digital Credentials, which verify authenticity like physical documents. Losing a physical document doesn't mean losing the privileges it represents; it just means the User can't prove them. However, if a User loses privileges due to a serious infraction, the Authentic Source will revoke the related attributes. In such cases, when a User's attributes are updated, Authentic Sources MUST notify Credential Issuers to update the validity status of any Digital Credential containing those attributes.

Authentic Sources using PDND MUST use Signal Hub as the only notification update mechanism. In this case, they MUST deposit a Signal through the *Signal Collection e-Service* (page 609) in the following cases:

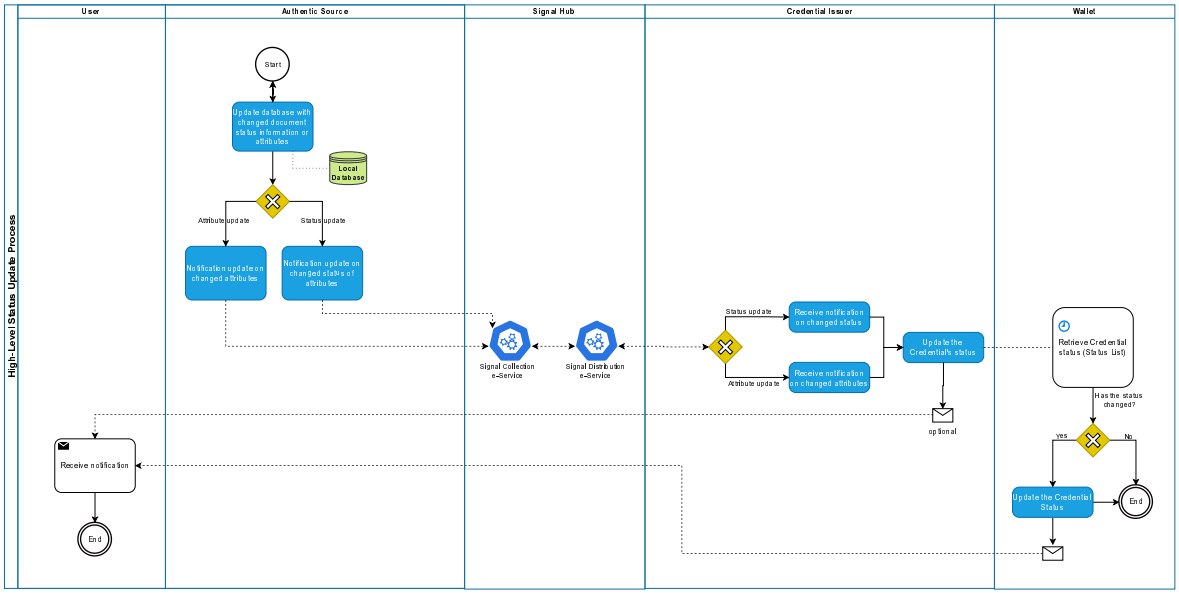
- The value of one or more Attributes contained in the Authentic Source's database has changed;
- The validity status of the Attributes is updated (revocation or suspension).

In both cases, the Signal MUST have `signalType` set to `UPDATE`.

Credential Issuers MUST check the PDND Signal Hub *Signal Distribution e-Service* (page 611) periodically for new Signals. For the Signal processing flow, please refer to the Section *Signals Processing* (page 613). The Credential Issuer is able to identify the nature of the `UPDATE` Signal

by querying the Authentic Source *get attribute* API and inspecting the response payload, as described in Section *Get Attribute Claims* (page 603).

The following diagram illustrates the high-level status update process for Authentic Sources.



The process starts with data or data validity changes that occur in the Authentic Source data. Changes can also be initiated by third-party entities other than the Authentic Source, such as when law enforcement agencies report illegal activities.

Once the data changes, the Authentic Source notifies the Credential Issuers who received the original data using the Signal Hub. The Authentic Source deposits a Signal in the Signal Collection e-Service. *Signal Collection e-Service* (page 609).

The Credential Issuer periodically queries the Signal Hub *Signal Distribution e-Service* (page 611) for new Signals. When a new Signal is found, the Credential Issuer retrieves it and processes it as described in *Signals Processing* (page 613). Then, the Credential Issuer updates the Credential Status according to the validity mechanism's defined mode. The Credential Issuer MAY notify the User through a registered out-of-band communication channel.

The Wallet instance, following periodic checks of the validity status of the stored Digital Credentials, receives the updated status. When the Credential Status is changed to `INVALID`, the Credential Issuer MUST inform the User about this change. In case the Credential status is modified to `UPDATE` (resp. 0x03) or `ATTRIBUTE_UPDATE` (resp. 0x0F), the Wallet Instance SHOULD proceed to the re-issuance of the Digital Credential, as described in *Re-Issuance Flow* (page 448).

Batch Credential Lifecycle Management

When multiple Digital Credentials are issued together in a single batch, their lifecycle remains fully granular:

- **Grouped triggers, independent updates**: regardless of the actor that triggers a batch status update (e.g. the Wallet Instance via Notification Endpoint with `event=credential_deleted`, Wallet Provider via updating Wallet Instance and Key Attestation status list) the status updating is handled as one or more separate status changes. The Credential Issuer updates each Credential's status individually (for instance, by flipping its status-list bit to `INVALID` or `SUSPENDED`). The Wallet Instance MUST NOT trigger batch status updates when the User deletes local Credentials. Upon deletion, the Wallet Instance MAY, under the User's explicit consent, notify the Credential Issuer of the User's intention to revoke the affected Credential(s).

Note

As the Wallet UI typically surfaces a batch as one Credential (e.g., 3 uses remaining), a User-driven deletion in the Wallet removes the entire batch locally. By default it does not request revocation at the Issuer. The Wallet MAY offer the User an optional prompt to request revocation at the Issuer as part of the deletion flow.

11.4.4 Validity Verification Mechanisms

For the verification of the validity status of a long-lived Digital Credential the Token Status List (TOKEN-STATUS-LIST) MUST be supported for both the remote and proximity scenario. The following table sums up the required revocation mechanisms for verifying the status of long-lived Digital Credentials.

Flow	Revocation Mechanism	Reference
Remote	• [REQUIRED] Token Status List.	TOKEN-STATUS-LIST.
Proximity	[REQUIRED] Token Status List.	TOKEN-STATUS-LIST.

Token Status Lists

This section defines a Status List data structure, which is used to convey information regarding the individual statuses of multiple Digital Credentials. Digital Credentials may be of any format, such as SD-JWTs or ISO/IEC 18013-5 mdocs. A Status List describes the status of the Digital Credentials by encoding their status validity in a bit array. Each Digital Credential is allocated an index during issuance; this index represents its position within the bit array. The value of the bit(s) at this index corresponds to the Digital Credentials' status. A Status List is provided within a cryptographically signed Status List Token in JWT format. For details, see TOKEN-STATUS-LIST.

In this specification, the roles of Credential Issuer and Status Issuer (i.e., the entity that issues the Status List Token about the status information of the Digital Credential) coincide, whereas the Status Provider (i.e., the entity that provides the Status List Token on a public endpoint) MAY be the Credential Issuer itself or another entity.

The Issuer of the Digital Credentials MUST

- define a number of bits, k, (either 1, 2, 4, 8) that represents the amount of bits used to describe the status of each Digital Credential within this Status List. The Credential Issuer MUST configure the number of bits. Each Credential will therefore have 2^k (where k is the number of bits chosen) possible states.
- create a byte array of size = (amount of Digital Credentials) * k / 8 or greater. Depending on k, each byte in the array corresponds to 8/k statuses (8 if k=1, 4 if k=2, 2 if k=1, or 1 if k=8). Each time a Digital Credential is issued the Credential Issuer assigns it to a position in the array.
- set the status values for all issued Digital Credentials within the byte array. The status of each Digital Credential is identified using an index that maps to one or more specific bits within the byte array. The index starts counting at 0 and ends with (amount of Digital Credential) - 1 (being the last valid entry). The bits within an array are counted from the least significant bit ("0") to the most significant bit ("7"). All bits of the byte array at a particular index are set to a status value.
- compress the byte array using DEFLATE [**RFC 1951**] with the ZLIB [**RFC 1950**] data format. Implementations are RECOMMENDED to use the highest compression level available.
- make available to Relying Parties, and Wallet Instances, an endpoint to request Status Lists.

The Issuer of a Digital Credential MUST use the following values for possible Statuses of the issued Digital Credentials:

- 0x00 - `VALID` - The Digital Credential is valid.

- 0x01 - `INVALID` - The Digital Credential is revoked.
- 0x02 - `SUSPENDED` - The Digital Credential is temporarily invalid, hanging. This state is usually temporary.
- 0x03 - `UPDATE` - The Digital Credential metadata parameters have changed.
- 0x0F - `ATTRIBUTE_UPDATE` - The Digital Credential attributes have changed.

For example, if five states for a certain Digital Credential are possible, then k=4. If the Credential Issuer creates an array to store the statuses of 6 Digital Credentials, whose validity statuses are 0, 0, 0, 3, 1, 2, respectively; it will:

- create the bit array `[0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 1, 1, 0, 0, 0, 0; 0, 0, 1, 0, 0, 0, 0, 1]` which in hexadecimal notation generates the byte array `[0x00, 0x30, 0x21]`.
- compress the array using DEFLATE.

Note

When the Credental Issuer choses the number of bits for conveying statuses of the Digital Credentials it issues, it MAY add other states besides those described above. The addition of many different states for the lifecycle of a Digital Credential has however to be carefully pondered for it discloses information to Relying Parties.

Note

The main privacy consideration for a Status List is to prevent the Issuer from tracking the usage of the Digital Credential when the status is being checked. If a Credential Issuer offers status information by referencing a specific token, this would enable the Credential Issuer to create a profile for the issued token by correlating the date and identity of Relying Parties, that are requesting the status. Implementations MUST therefore integrate the status information of many Digital Credentials into the same list. As a result, the Issuer does not learn for which Digital Credential the Relying Party is requesting the Status List. The privacy of the User is protected by the anonymity within the set of Digital Credential in the Status List, this limits the possibilities of tracking by the Issuer. This herd privacy effect depends on the number of entities within the Status List. A larger amount of Digital Credentials referenced therein results in better privacy but also impacts the performance as more data has to be transferred to read the Status List. Depending on the Status List parameters (e.g. the amounts of bits designating the Credential values), Credential Issuers have to strike an appropriate balance between privacy and performance.

Once the Relying Party receives a Digital Credential, this enables it to request the Status List to validate its status through the provided URI parameter and look up the corresponding index.

However, the Relying Party is able to store the URI and index of the Digital Credential to request the Status List again at a later time. By doing so regularly, the Relying Party may create a profile of the Digital Credential's validity status. This behaviour might also be abused in cases where this is not intended and unknown to the User, e.g. profiling the suspension of a driving license. This behaviour could be mitigated e.g., by regular re-issuance of the Digital Credential.

Status List Token

The Status List Token is available at the Status List Endpoint and contains the following parameters.

Header	Description	Reference
alg	REQUIRED. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or to a symmetric algorithm (MAC) identifier.	[**RFC 7515**], [**RFC 7517**].
typ	REQUIRED. It MUST be set to `statuslist+jwt`.	TOKEN-STATUS-LIST
kid	REQUIRED. Unique identifier of the Credential Issuer's public key which signs the Status Token.	**RFC 7638#section_3**.
x5c	REQUIRED. X.509 public key certificate or certificate chain corresponding to the key used to sign the Status List Token.	**RFC 5280**

Payload	Description	Reference
sub	REQUIRED. The subject claim MUST specify the URI of the Status List Token. The value MUST be equal to that of the `uri` claim contained in the `status_list` claim of the Digital Credential.	[**RFC 7519**]
iat	REQUIRED. The issued at claim MUST specify the time at which the Status List Token was issued.	[**RFC 7519**]
exp	RECOMMENDED. The expiration time claim, if present, MUST specify the time at which the Status List Token is considered expired by the Credential Issuer.	[**RFC 7519**]

continues on next page

Table 11.18 – continued from previous page

Payload	Description	Reference
ttl	RECOMMENDED. The time to live claim, if present, MUST specify the maximum amount of time, in seconds, that the Status List Token can be cached by a consumer before a fresh copy SHOULD be retrieved. The value of the claim MUST be a positive number encoded in JSON as a number. This amount of time SHOULD NOT exceed the expiration time defined in **exp** claim.	TOKEN-STATUS-LIST
status_list	REQUIRED. JSON Object that contains a Status List.	TOKEN-STATUS-LIST

Note

It is RECOMMENDED that the Credential Issuer sets the `exp` claim so that the Status List Token is short-lived. Typically, this involves the `exp` claim not to exceed the `iat` claim by more than 24 hours.

A JSON-encoded Status List has the following structure:

Parameter	Description	Reference
bits	REQUIRED. JSON Integer specifying the number of bits per Digital Credential in the compressed byte array (*lst*). The allowed values for bits are 1,2,4 and 8.	TOKEN-STATUS-LIST
lst	REQUIRED. JSON String that contains the status values for all the Digital Credentials it conveys statuses for. The value MUST be the base64url-encoded compressed byte array.	TOKEN-STATUS-LIST
aggregation_uri	OPTIONAL. JSON String that contains a URI to retrieve the Status List Aggregation for this type of Digital Credential or Issuer.	TOKEN-STATUS-LIST

The following is an example of Status List Token before applying signature and encoding:

```
{
  "alg": "ES256",
  "kid": "$KID",
  "typ": "statuslist+jwt",
  "x5c": [
    "MIIDqjCCApKgAwIBAgIESLNEvDA ...",
    "MIICwzCCAasCCQCKVy9eKjvi+jA ...",
    "MIIDTDCCAjSgAwIBAgIJAPlnQYH..."
  ]
}
```

```
{
  "exp": 2291720170,
  "iat": 1686920170,
  "status_list": {
    "bits": 1,
    "lst": "eNrbuRgAAhcBXQ"
  },
  "sub": "https://example-issuer.com/statuslists/",
  "ttl": 43200
}
```

Handling Credential Status with Status List Token

Credential Issuers, once a Digital Credential has been generated, MUST:

- Store it locally with minimum set of data required to manage its lifecycle, including the validity status of that Digital Credential;
- Include a `status_list` claim within the JSON Object value of the `status` claim of the Digital Credential.

The value of the claim `status_list` MUST be itself a JSON Object with the following parameters

Parameter	Description	Reference
idx	REQUIRED. The idx (index) claim MUST specify an Integer that represents the index to check for status information in the Status List for the current Digital Credential. The value of idx MUST be a non-negative number, containing a value of zero or greater.	TOKEN-STATUS-LIST
uri	REQUIRED. The uri (URI) claim MUST specify a String value that identifies the Status List Token containing the status information for the Digital Credential. The value of uri MUST be a URI conforming to [**RFC 3986**].	TOKEN-STATUS-LIST

Checking Credentials Statuses

The fetching, processing and verifying of a Status List Token may be done by either the Wallet Instance or a Relying Party. Below it is described for the Wallet Instance, however, the same rules would also apply to the Relying Party.

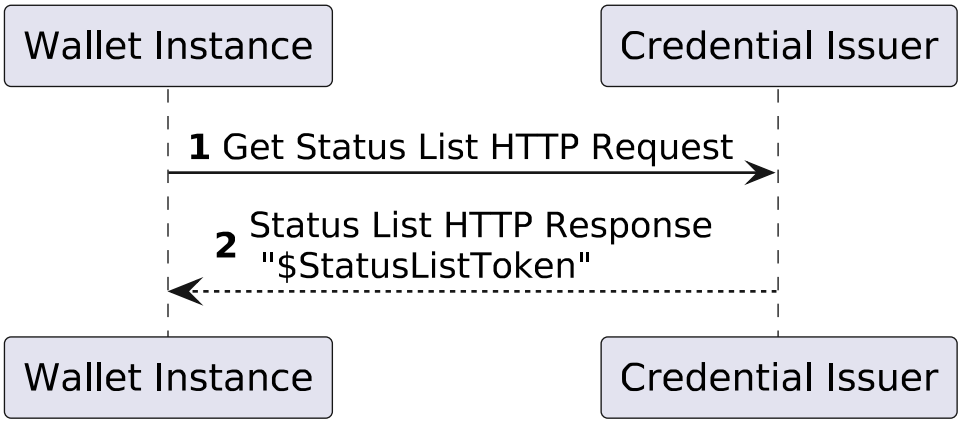


Fig. 11.4: Status List Flow.

HTTP Status List Request

To obtain the Status List Token, the Relying Party MUST send an HTTP GET request to the `status.status_list.uri` value provided within the Digital Credential.

The Relying Party SHOULD send the `application/statuslist+jwt` Accept-Header to indicate that the requested response type for Status List Token is the JWT format.

The following is a non-normative example of a request for a Status List Token:

```
GET /statuslists HTTP/1.1
Host: example-issuer.com
Accept: application/statuslist+jwt
```

HTTP Status List Response

The Status List Endpoint responds with a Status List Token and MUST use an HTTP status code in the 2xx range. In the successful response, the Status Provider MUST use content-type `application/statuslist+jwt` for Status List Token in JWT format.

The HTTP response SHOULD use gzip Content-Encoding as defined in [**RFC 9110**].

If caching-related HTTP headers are present in the HTTP response, Relying Parties SHOULD prioritize the `exp` and `ttl` claims within the Status List Token over the HTTP headers for determining caching behavior.

The following is a non-normative example of a response for a Status List Token with type `application/statuslist+jwt`:

```
HTTP/1.1 200 OK
Content-Type: application/statuslist+jwt
```

eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyIiwidHlwIjoic3RhdHVzbGlzdCtqd3QifQ.eyJleHAiOjIyOTE3MjAxNzAsIml
hdCI6MTY4NjkyMDE3MCwiaXNzIjoiaHR0cHM6Ly9leGFtcGxlLmNvbSIsInN0YXR1c19saXN0Ijp7ImJpdHMiOjEsImxzd
CI6ImVOcmJ1UmdBQWhjQlhRIn0sInN1YiI6Imh0dHBzOi8vZXhhbXBsZS5jb20vc3RhdHVzbGlzdHMvMSIsInR0bCI6NDM
yMDB9.SSdg3AnTHsyRtCHziLy-QnXg-YRldMEXkdEgDXgE_ZvIvjM0eULQlzEbLBLfCeGhlqKJSReC-m85K79CTjJDzg

Upon receiving a Digital Credential, a Relying Party MUST first perform the validation of the Digital Credential itself (e.g., checking for expected attributes, valid signature and expiration time). If this validation is not successful, the Digital Credential MUST be rejected. If the validation was successful, the Relying Party MUST perform the following validation steps to evaluate the status of the Digital Credential:

- Check for the existence of a `status` claim, check for the existence of a `status_list` claim within the `status` claim and validate that the content of `status_list` adheres to the rules defined in Section *Handling Credential Status with Status List Token* (page 415).
- Resolve the Status List Token from the provided URI.
- Validate the Status List Token:
  - Validate the Status List Token's signature by following the rules defined in section 7. 2 of [**RFC 7519**]. This step requires the resolution of a public key as described in *The Infrastructure of Trust* (page 109).
  - Check for the existence of the required claims as defined in Section *Status List Token* (page 413).
- All existing claims in the Status List Token MUST be checked according to *Status List Token* (page 413).
  - The subject claim of the Status List Token MUST be equal to the `uri` claim in the `status_list` object of the Digital Credental.
  - If the Relying Party has custom policies regarding the freshness of the Status List Token, it SHOULD check the `iat` claim.
  - If the expiration time is defined, it MUST be checked if the Status List Token is expired.
  - If the Relying Party is using a system for caching the Status List Token, it SHOULD check the `ttl` claim of the Status List Token and retrieve a fresh copy if (time status was resolved + `ttl` < current time).
- Decompress the Status List with a decompressor that is compatible with DEFLATE [**RFC 1951**] and ZLIB [**RFC 1950**].
- Retrieve the status value of the index specified in the Digital Credential as described in *Checking Credentials Statuses* (page 415). Fail if the provided index is out of bounds of the Status List.

- Check the status value as described in *Checking Credentials Statuses* (page 415).

If any of these checks fails, no statement about the status of the Digital Credential can be made and the Digital Credential SHOULD be rejected.

If for example, the decompressed byte array is `[0x00, 0x40, 0x21]`, it corresponds to the bit array `[0, 0, 0, 0, 0, 0, 0, 0; 0, 1, 0, 0, 0, 0, 0, 0; 0, 0, 1, 0, 0, 0, 0, 1]`. The Status of the Digital Credential whose `idx` claim value is `5` in this array refers to the last 4-bit pair (i.e., `[0, 0, 1, 0] = 0x02`) whose status value is `SUSPENDED`.

In case any error occurs when the Status Token Endpoint generates the response, following HTTP Status Codes MUST be supported:

Status Code	Description
500 Internal Server Error [REQUIRED]	The Status List Provider encountered an internal problem.
503 Service Unavailable [REQUIRED]	The Status List Provider is temporary unavailable.
504 Gateway Timeout [OPTIONAL]	The Status List Provider cannot fulfill the request within the defined time interval.

CHAPTER

TWELVE

DIGITAL CREDENTIAL FLOWS

12.1 Digital Credential Issuance

This section describes the PID and (Q)EAAs issuance flow with a high level of security.

12.1.1 Credential Issuance High-Level Flows

High-Level PID flow

The Fig. 12.1 shows a general architecture and highlights the main operations involved in the issuance of a PID.

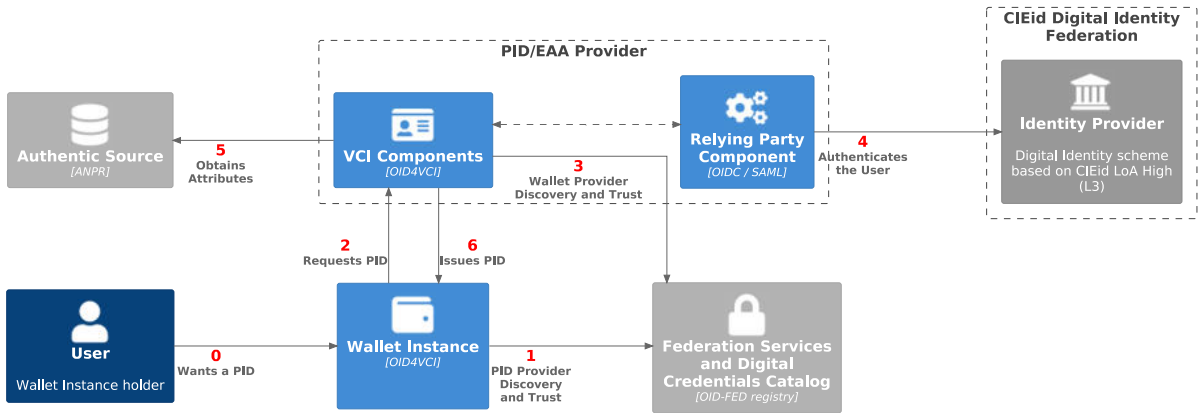


Fig. 12.1: PID Issuance - General architecture and high level flow.

The high-level flow begins with the User who wants to obtain a PID and starts his/her Wallet Instance (Step 0). Below the description of the steps represented in the previous picture:

1. **PID Provider Discovery and Trust**: the Wallet Instance discovers the trusted PID Provider using the Digital Credential Catalogue and Federation Services, establishing the trust to the PID Provider according to the Trust Model and obtaining its metadata that discloses the formats of the PID, the algorithms supported, and any other parameter required for interoperability needs (*WP_045–046* (page 746)).

2. **PID Request**: using the Authorization Code Flow defined in [OpenID4VCI] the Wallet Instance requests the PID to the PID Provider (*WP_051* (page 746)).
3. **Wallet Provider Discovery and Trust**: the PID Provider checks the authenticity and validity of the Wallet Instance, establishing the trust to the Wallet Provider and obtaining Wallet metadata with the parameters required for interoperability needs, according to the Trust Model.
4. **User Authentication**: the PID Provider authenticates the User according to credential-issuance-endpoint:User Authentication Method Selection.
5. **Fetch of PID data from National Public Registry**: the PID Provider obtains the required PID data from National Public Registry (ANPR) which acts as Authentic Source.
6. **PID Issuance**: the PID Provider releases a PID bound to the key material held by the requesting Wallet Instance.

High-Level IT-Wallet ID flow

The Fig. 12.2 shows a general architecture and highlights the main operations involved in the issuance of an IT-Wallet ID.

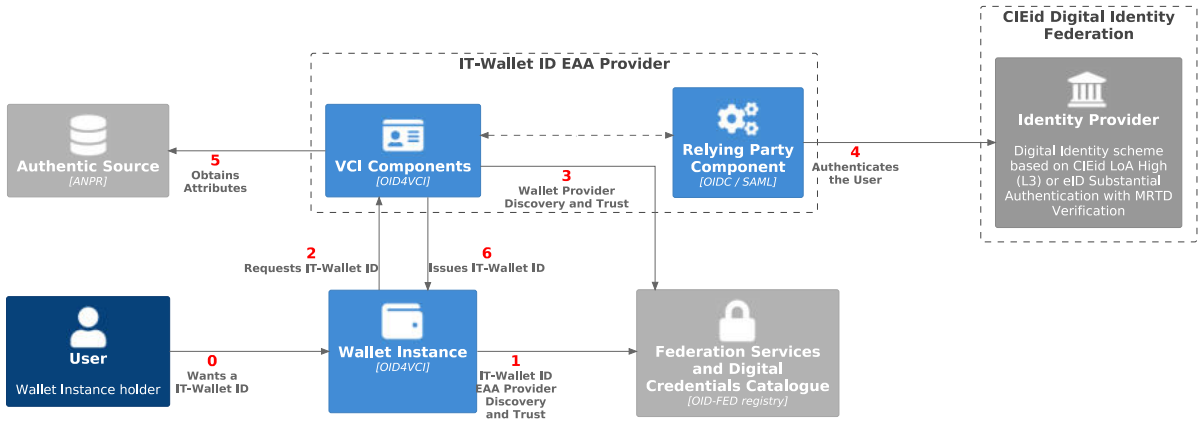


Fig. 12.2: IT-Wallet ID Issuance - General architecture and high level flow.

The high-level flow begins with the User who wants to obtain a IT-Wallet ID and starts his/her Wallet Instance (Step 0). Below the description of the steps represented in the previous picture:

1. **IT-Wallet ID Provider Discovery and Trust**: the Wallet Instance discovers the trusted IT-Wallet ID EAA Provider using the Digital Credential Catalogue and Federation Services, establishing the trust to the IT-Wallet EAA Provider according to the Trust Model and obtaining its metadata that discloses the formats of the IT-Wallet ID, the algorithms supported, and any other parameter required for interoperability needs (*WP_045-046* (page 746)).

2. **IT-Wallet ID Request**: using the Authorization Code Flow defined in [OpenID4VCI] the Wallet Instance requests the IT-Wallet ID to the EAA Provider (*WP_051* (page 746)).
3. **Wallet Provider Discovery and Trust**: the IT-Wallet ID EAA Provider checks the authenticity and validity of the Wallet Instance, establishing the trust to the Wallet Provider and obtaining Wallet metadata with the parameters required for interoperability needs, according to the Trust Model.
4. **User Authentication**: the IT-Wallet ID EAA Provider authenticates the User according to credential-issuance-endpoint:User Authentication Method Selection. For the L2+ protocol flow, see *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457).
5. **Fetch of IT-Wallet ID data from National Public Registry**: the IT-Wallet ID EAA Provider obtains the required IT-Wallet ID data from National Public Registry (ANPR) which acts as Authentic Source.
6. **IT-Wallet ID Issuance**: the IT-Wallet ID EAA Provider releases a IT-Wallet ID bound to the key material held by the requesting Wallet Instance.

High-Level (Q)EAA flow

The Fig. 12.3 shows a general architecture and highlights the main operations involved in the issuance of a (Q)EAA, following the assumptions listed below:

- the User has a valid PID stored in their own Wallet Instance, or a valid IT-Wallet ID when requested according to credential-issuance-endpoint:User Authentication Method Selection;
- the (Q)EAA requires a high security implementation profile.

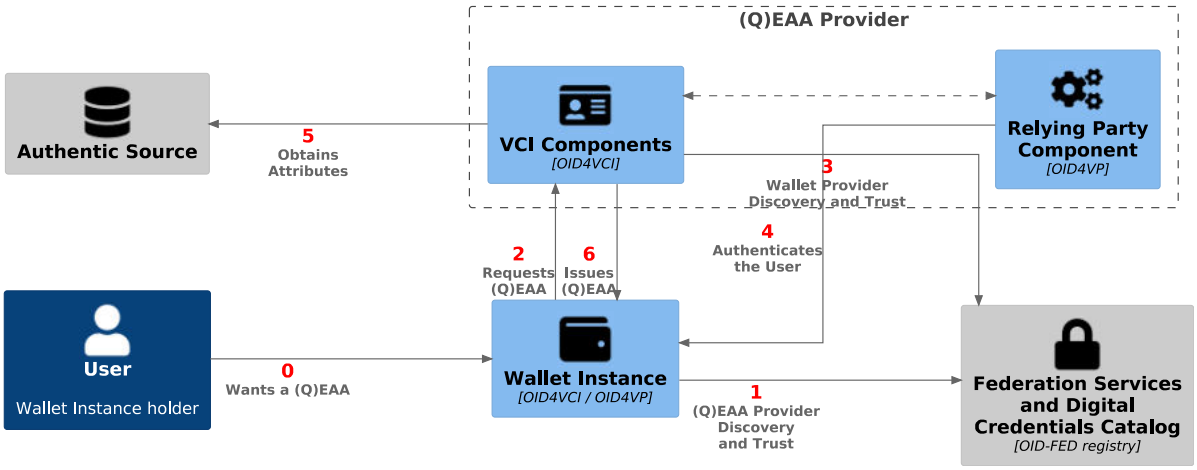


Fig. 12.3: (Q)EAA Issuance - General architecture and high level flow.

Similarly to the PID and IT-Wallet ID high-level flow, the above diagram depicts a (Q)EAA high-level flow starting from the User who wants to obtain a (Q)EAA (step 0). Below the description of the most relevant operations involved in the (Q)EAA issuance:

1. **(Q)EAA Provider Discovery and Trust**: the Wallet Instance obtains the list of the trusted (Q)EAA Providers using the Digital Credential Catalogue and Federation API (e. g.: using the Subordinate Listing Endpoint of the Trust Anchor and its Intermediates), then inspects the metadata looking for the Digital Credential capabilities of each (Q)EAA Provider (*WP_045–046* (page 746)).
2. **(Q)EAA Request**: using the Authorization Code Flow, defined in [OpenID4VCI], the Wallet Instance requests a (Q)EAA to the (Q)EAA Provider (*WP_051* (page 746)).
3. **Wallet Provider Discovery and Trust**: the (Q)EAA Provider verifies the authenticity and validity of the Wallet Instance. During this step the (Q)EAA Provider establishes trust with the Wallet Provider and retrieves Wallet metadata containing the necessary parameters for interoperability, as defined by the Trust Model.
4. **User Authentication**: the (Q)EAA Provider, acting as a Relying Party Instance, authenticates the User according to credential-issuance-endpoint:User Authentication Method Selection.
5. **Obtaining Attributes**: the (Q)EAA Provider fetches User attributes from the relevant Authentic Source.
6. **(Q)EAA Issuance**: the (Q)EAA Provider releases a (Q)EAA bound to the key material held by the requesting Wallet Instance.

12.1.2 Credential Issuance Low-Level Flows

Low-Level Issuance Flow

The Credential Issuance flow is based on [OpenID4VCI] and the following main reference standards/specifications MUST be supported on top of OpenID4VCI:

- **The OAuth 2.0 Authorization Framework** [**RFC 6749**], as recommended in Section 3 of [OpenID4VCI].
- **Pushed Authorization Requests** (PAR) [**RFC 9126**], as recommended in Section 5 of [OpenID4VCI].
- **Proof Key for Code Exchange** (PKCE) [**RFC 7636**], as recommended in Section 5 of [OpenID4VCI].
- **JWT Authorization Requests** (JAR) [**RFC 9101**].
- **Rich Authorization Requests** (RAR) [**RFC 9396**].

- **OAuth 2.0 Attestation-Based Client Authentication** [OAUTH-ATTESTATION-CLIENT-AUTH].
- **OpenID Federation 1.0** [OID-FED].

The Credential Issuer MUST use *OAuth 2.0 Authorization Server* based on **RFC 6749** to authorize the User to obtain a Credential. Credential Issuers MUST support:

- **Authorization Code Flow**: The Credential Issuer requires User authentication and consent at the Authorization Endpoint before collecting User information to create and provide a Credential.
- **Wallet Initiated Flow**: The request from the Wallet Instance is sent to the Credential Issuer without any input from the Credential Issuer or a third party (i.e. via Credential Offer support).
- **Immediate Issuance Flow**: The Credential Issuer issues the Credential directly in response to the Credential Request.

In addition, the Credential Issuers MAY support:

- **Third-Party Initiated Flow**: The Wallet Instance sends its request to the Credential Issuer based on the input provided by the Credential Issuer or a third party (for example an Authentic Source) supporting Credential Offer mechanism as defined in [OpenID4VCI].
  - **Same-device Issuance Flow**: The User receives the Credential on the same device used to initiate the flow.
  - **Cross-device Issuance Flow**: The User receives the Credential on another device than the one that initiated the flow.
- **Refresh Token Flow**: The Wallet Instance requests a new Access Token at the Token Endpoint of the PID/(Q)EEA Provider.
- **Re-issuance Flow**: Following updates to an already stored Digital Credential, the Wallet Instance requests a refresh of the Digital Credential at the Credential Endpoint of the Credential Issuer.
- **Deferred Issuance Flow**: The Credential Issuer may require time to issue the requested Digital Credential, due to the Authentic Sources data provisioning rules, and allows the Wallet to retrieve the requested Credential in the future.
- **Batch Credential Issuance Flow**: It enables the issuance of a batch of one or more Digital Credential. Digital Credentials that are issued in a batch MUST share the same format and contain the same set of attributes about the Holder. Each Credential MUST contain different Cryptographic Data to achieve unlinkability between the Digital Credentials.

Note

Standard or Batch Credential Issuance:

The User can configure the Wallet Solution to issue Digital Credentials in either batch or standard mode and define the preferred batch size.

The entire Issuance flow can be divided into two sub-flows:

- **User Request Flow**, describing the modes through which the User can request the Credential. It can be:
  1. On the initiative of the User (**Wallet Initiated**).
  2. Upon proposal of the Credential Issuer (**Issuer Initiated**).
- **Issuance Flow**, describing interactions between Wallet Instance and Credential Issuer.

The following diagram shows the *User request flow*.

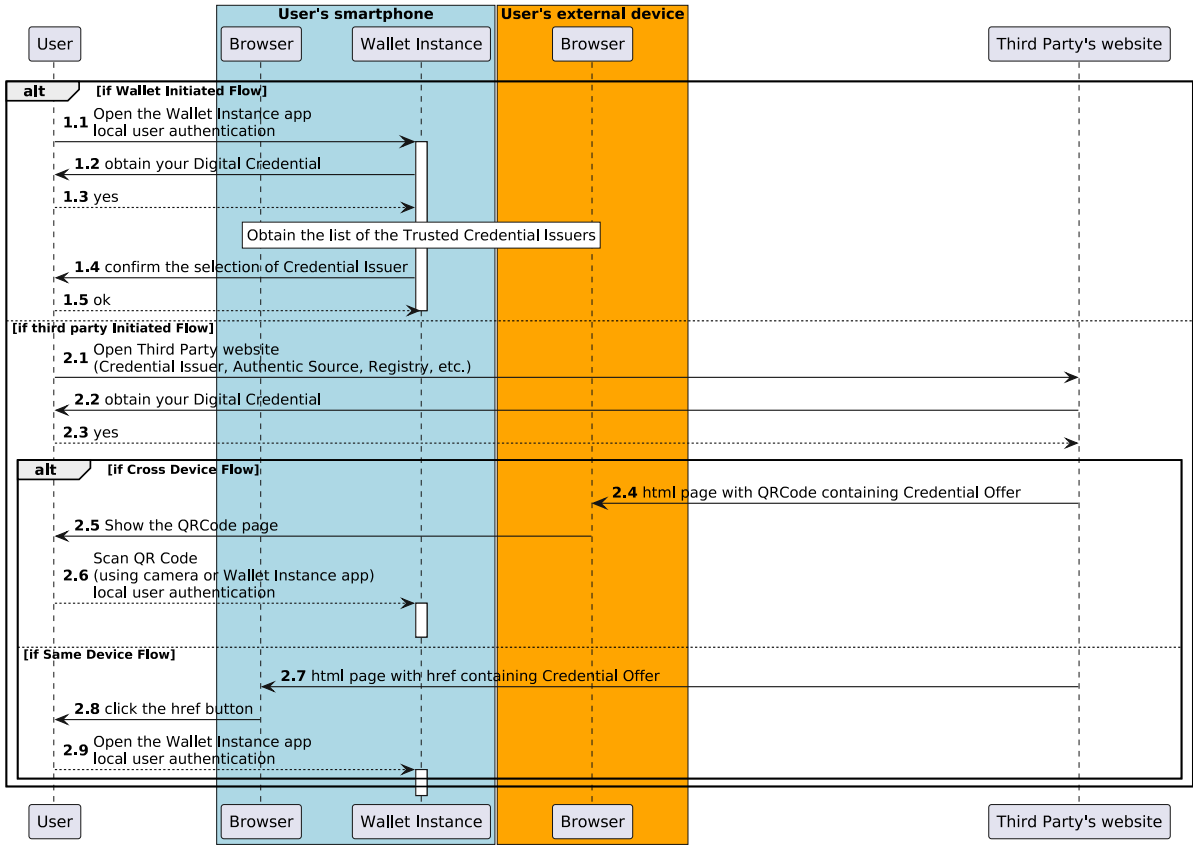


Fig. 12.4: Credential Request by the User - Detailed flow.

Steps 1.1-1.4 (Wallet Initiated Flow): The User, using the Wallet Instance, selects the Credential Issuer from those listed in the list of trustworthy entities.

Steps 2.1-2.3 (Third Party Initiated Flow): The User while browsing the Credential Issuer website or a Third Party website supporting the Credential Offer mechanism (for example an Authentic Source) finds a link to obtain a Digital Credential.

Steps 2.4-2.6 (Cross-Device): The Credential Offer is presented as a QR Code. The User scans it with the device camera or with the Wallet Instance's built-in scanner, triggering the Wallet Instance to retrieve the parameters defined in the *Table of Credential Offer parameters* (page 453) (*WP_047-048* (page 746)).

Steps 2.7-2.10 (Same-Device): The Credential Offer is presented as an href button containing the URL that allows the User to invoke the Wallet Instance using the Credential Offer Endpoint.

The Credential Offer can be sent by value (using the `credential_offer` parameter) or by reference (using the `credential_offer_uri` parameter) as defined in Section 4 of [OpenID4VCI]. Additional details and non-normative examples are provided in Section *Credential Offer Flow* (page 452).

The following diagram shows the *Issuance flow*.

Once *User Request flow* is completed, the Wallet Instance processes the Metadata of the Credential Issuer as defined in Section *Trust Evaluation Mechanism* (page 131). Additionally, in the case of Batch Credential issuance, the Wallet Instance MUST check the support of batch issuance by looking for the `batch_credential_issuance` object in the Credential Issuer metadata, from where the Wallet Instance can get the `batch_size` value. Batch issuance applies only to the Credential types for which that metadata is advertised. The Credential Issuer MUST declare whether PID and IT-Wallet ID are in scope. A Credential type without that advertisement MUST be issued as a single Credential (*CI_203* (page 776)).

Note

Federation Check: The Wallet Instance must verify whether the Credential Issuer is a member of the Federation, obtaining its protocol specific Metadata (*WP_046* (page 746)). A non-normative example of a response from the endpoint **.well-known/openid-federation** with the **Entity Configuration** and the **Metadata** of the Credential Issuer is represented within the section credential-issuer-entity-configuration:Credential Issuer Entity Configuration.

In case of Issuer Initiated flow, in addition to the Federation Check defined above, the Wallet Instance MUST execute the following checks on the Credential Offer parameters:

- For each Credential identifier contained in the `credential_configuration_ids` array verify if it is supported by the Credential Issuer (*WP_050* (page 746)).
- The Authorization Server identifier (if present) is contained in the `authorization_servers` Credential Issuer metadata parameter (*WP_049* (page 746)).

Steps 1-2 (PAR Request): The Wallet Instance:



Fig. 12.5: Credential Issuance - Detailed flow.

- Creates a fresh PKCE code verifier, Wallet Instance Attestation Proof of Possession, and `state` parameter for the *Pushed Authorization Request* (*WP_052* (page 746)).
- Provides to the Credential Issuer PAR endpoint the parameters previously listed above, using the `request` parameter (hereafter Request Object) according to **RFC 9126** Section 3 to prevent Request URI swapping attack (*WP_052* (page 746)). The Pushed Authorization Request enables client authentication prior to any User interaction. This step allows for the early rejection of illegitimate requests, effectively preventing spoofing attacks, tampering, and improper use of authorization requests.
- MUST create the `code_verifier` with enough entropy random string using the unreserved characters with a minimum length of 43 characters and a maximum length of 128 characters, making it impractical for an attacker to guess its value. The value MUST be generated following the recommendation in Section 4.1 of **RFC 7636** (*WP_052a* (page 746)).
- Signs this request using the private key that is created during the setup phase to obtain the Wallet Instance Attestation. The related public key that is attested by the Wallet Provider is provided within the Wallet Instance Attestation `cnf.jwk` claim (*WP_052c* (page 746)).
- MUST use the `OAuth-Client-Attestation` and `OAuth-Client-Attestation-PoP` parameters according to OAuth 2.0 Attestation-based Client Authentication [OAUTH-ATTESTATION-CLIENT-AUTH], since in this flow the Pushed Authorization Endpoint is a protected endpoint (*WP_052b* (page 746)).
- MUST specify the types of the requested credentials using both the `scope` parameter, as required by [OPENID4VC-HAIP], and the `authorization_details` [RAR **RFC 9396**] parameter (*WP_052d* (page 746)).

Note

This specification uses JAR [**RFC 9101**] to ensure end-to-end integrity of authorization requests for all Request Object parameters. When interacting with cross-border Credential Issuers whose Authorization Servers may not support JAR, Wallet Instances SHOULD check the `require_signed_request_object` parameter in the Authorization Server metadata to determine whether to sign the Request Object. Wallet Solutions supporting both JAR-compliant and non-compliant Authorization Servers may duplicate parameters in both the request body and the signed Request Object. Section 10.7 of **RFC 9101** provides the security requirements on how to manage this duplication properly.

Note

User authentication method selection, including the use of `it_l2+document_proof`, is defined in credential-issuance-endpoint:User Authentication Method Selection. The related protocol

flow is defined in *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457).

The Credential Issuer performs the following checks upon the receipt of the PAR request:

1. It MUST validate the signature of the Request Object using the algorithm specified in the `alg` header parameter (**RFC 9126**, **RFC 9101**) and the public key retrieved from the Wallet Instance Attestation (`cnf.jwk`) referenced in the Request Object, using the `kid` JWT header parameter.
2. It MUST check that the used algorithm for signing the request in the `alg` header is one of the listed within the Section *Cryptographic Algorithms* (page 615).
3. It MUST check that the `client_id` in the request body of the PAR request matches the `client_id` claim included in the Request Object.
4. It MUST check that the `iss` claim in the Request Object matches the `client_id` claim in the Request Object (**RFC 9126**, **RFC 9101**).
5. It MUST check that the `aud` claim in the Request Object is equal to the identifier of the Credential Issuer (**RFC 9126**, **RFC 9101**).
6. It MUST reject the PAR request, if it contains the `request_uri` parameter (**RFC 9126**).
7. It MUST check that the Request Object contains all the mandatory parameters which values are validated according to *Table of the HTTP parameters* (page 566) [derived from **RFC 9126**].
8. It MUST check that the Request Object is not expired, checking the `exp` claim.
9. It MUST check that the Request Object was issued in a previous time than the value exposed in the `iat` claim. It SHOULD reject the request if the `iat` claim is far from the current time (**RFC 9126**) of more than *5* minutes.
10. It MUST check that the `jti` claim in the Request Object has not been used before by the Wallet Instance identified by the `client_id`. This allows the Credential Issuer to mitigate replay attacks (**RFC 7519**).
11. It MUST validate the `OAuth-Client-Attestation-PoP` parameter based on Section 5 of [OAUTH-ATTESTATION-CLIENT-AUTH].
12. It MUST verify, if present, the validity of the `issuer_state` and the coherence with the `credential_configuration_id` requested in the Request Object.

Below is a non-normative example of the PAR Request.

```
POST /as/par HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
```

```
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sIndhbGxldF9uYW1lIjoiV2Fsb
GV0X3YxIiwid2FsbGV0X2xpbmsiOiJodHRwczovL2V4YW1wbGUuY29tL3dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2Fs
bGV0X3ZlcnNpb24iOiIxLjAuMCIsImNsaWVudF9zdGF0dXMiOnsic3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTM
zNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3RhdHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH
0sImV4cCI6MTc0MDE1ODE2N30.j3nx-adTbi3_O0Seddi_-Fqr37MJpDHCI5leCie7iTXApqn_4kOlrDgpmY35pGfZhlB_1
lZUBTnKywNn5LjNow
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA

client_id=$thumbprint-of-the-jwk-in-the-cnf-wallet-instance-attestation$&
request=$SIGNED-JWT
```

Below is a non-normative example of the Wallet Instance Attestation Proof of Possession (WIA-PoP) header and body:

```
{
    "typ": "oauth-client-attestation-pop+jwt",
    "alg": "ES256"
}
```

```
{
    "iss": "47b982369791d08003a7283f059cb0d1",
    "aud": "https://eaa-provider.example.org",
    "iat": 1715842560,
    "jti": "f8555ceb-c65c-4025-9378-b6672b6149af"
}
```

Below is a non-normative example of the signed Request Object without encoding and signature applied:

```
{
    "typ": "oauth-authz-req+jwt",
    "alg": "ES256",
    "kid": "b01b8208d9e6cc834d87dc356ab50170"
}
```

```
{
    "jti": "f8555ceb-c65c-4025-9378-b6672b6149af",
    "aud": "https://eaa-provider.example.org",
    "iat": 1715842560,
    "exp": 1715842860,
```

```
    "response_type": "code",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "iss": "47b982369791d08003a7283f059cb0d1",
    "state": "fyZiOL9Lf2CeKuNT2JzxiLRDink0uPcd",
    "code_challenge": "E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM",
    "code_challenge_method": "S256",
    "scope": "Education_degree",
    "authorization_details": [
        {
            "type": "openid_credential",
            "credential_configuration_id": "dc_sd_jwt_Education_degree"
        }
    ],
    "redirect_uri": "https://client.example.com/cb"
}
```


Note

Federation Check: The Credential Issuer MUST check that the Wallet Provider is part of the federation.

Note

The Credential Issuer MUST validate the signature of the Wallet Instance Attestation and that it is not expired.

Step 3 (PAR Response): The Credential Issuer provides a one-time use `request_uri` value. The issued `request_uri` value MUST be bound to the client identifier (`client_id`) that was provided in the Request Object.

Note

The entropy of the `request_uri` MUST be sufficiently large. The adequate shortness of the validity and the entropy of the `request_uri` depends on the risk calculation based on the value of the resource being protected. The validity time SHOULD be less than a minute, and the `request_uri` MUST include a cryptographic random value of 128 bits or more (**RFC 9101**). The entire `request_uri` SHOULD NOT exceed 512 ASCII characters due to the following two main reasons (**RFC 9101**):

1. Many phones on the market still do not accept large payloads. The restriction is typically either 512 or 1024 ASCII characters.

2. On a slow connection such as a 2G mobile connection, a large URL would cause a slow response; therefore, the use of such is not advisable from the user-experience point of view.

The Credential Issuer returns the issued `request_uri` to the Wallet Instance. A non-normative example of the response is shown below.

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json

{
    "request_uri": "urn:ietf:params:oauth:request_uri:bwc4JK-ESC0w8acc191e-Y1LTC2",
    "expires_in": 60
}
```

Steps 4-5 (Authorization Request): The Wallet Instance sends an authorization request to the Credential Issuer Authorization Endpoint (*WP_053* (page 746)). Since parts of this Authorization Request content, e.g., the `code_challenge` parameter value, are unique to a particular Authorization Request, the Wallet Instance MUST use a `request_uri` value once (**RFC 9126**) as per *WP_053a* (page 746). The Credential Issuer performs the following checks upon the receipt of the Authorization Request:

1. It MUST treat `request_uri` values as one-time use and MUST reject an expired request. However, it MAY allow for duplicate requests due to a User reloading/refreshing their user-agent (derived from **RFC 9126**).
2. It MUST identify the request as a result of the submitted PAR (derived from **RFC 9126**).
3. It MUST reject all the Authorization Requests that do not contain the `request_uri` parameter as the PAR is the only way to pass the Authorization Request from the Wallet Instance (derived from **RFC 9126**).

```
GET /authorize?client_id=$thumprint-of-the-jwk-in-the-cnf-wallet-instance-attestation$&request_
uri=urn%3Aietf%3Aparams%3Aoauth%3Arequest_uri%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: eaa-provider.example.org
```

Note

User Authentication and Consent: The Credential Issuer performs User authentication at the Authorization Endpoint according to credential-issuance-endpoint:User Authentication Method Selection. During this step, Credential Issuers MAY ask the User's contact details (e.g., their email address) to send notifications about the issued Digital Credential(s).

Steps 6-7 (Authorization Response): The Credential Issuer sends an authorization `code` together with `state` and `iss` parameters to the Wallet Instance. The Wallet Instance performs the following checks on the Authorization Response:

1. It MUST check the Authorization Response contains all the defined parameters according to *Table of the HTTP Response parameters* (page 570) (*WP_054* (page 746)).
2. It MUST check the returned value by the Credential Issuer for `state` parameter is equal to the value sent by Wallet Instance in the Request Object (**RFC 6749**) as per *WP_054a* (page 746).
3. It MUST check that the URL of Credential Issuer in `iss` parameter is equal to the URL identifier of intended Credential Issuer that the Wallet Instance starts the communication with (**RFC 9027**) as per *WP_054b* (page 746).

Note

The Wallet Instance redirect URI is a universal or app link registered with the local operating system, so this latter will resolve it and pass the response to the Wallet Instance.

```
HTTP/1.1 302 Found
Location: https://start.wallet.example.org?code=SplxlOBeZQQYbYS6WxSbIA&state=fyZiOL9Lf2CeKuNT2J
zxiLRDink0uPcd&iss=https%3A%2F%2Feaa-provider.example.org
```

Steps 8-9 (DPoP Proof for Token Endpoint): The Wallet Instance MUST create a new key pair for the DPoP and a fresh DPoP Proof JWT following the instruction provided in the Section 4 of (**RFC 9449**) for the token request to the Credential Issuer (*WP_055b* (page 746)). The DPoP Proof JWT is signed using the private key for DPoP created by Wallet Instance for this scope (*WP_055c* (page 746)). DPoP binds the Access Token, and optionally the Refresh Token, to a certain Wallet Instance (**RFC 9449**) and mitigates the misuse of leaked or stolen tokens at the Credential Endpoint.

Step 10 (Token Request): The Wallet Instance sends a token request to the Credential Issuer Token Endpoint with a *DPoP Proof JWT* and the parameters: `code`, `code_verifier`, and OAuth 2.0 Attestation based Client Authentication (`OAuth-Client-Attestation` and `OAuth-Client-Attestation-PoP`) as per *WP_055* (page 746).

The `OAuth-Client-Attestation` is signed using the private key bound to the Wallet Instance (*WP_055a* (page 746)). The related public key that is attested by the Wallet Provider is provided within the Wallet Instance Attestation (`cnf.jwk` claim) as per *WP_055d* (page 746). The Credential Issuer performs the following checks on the Token Request:

1. It MUST ensure that the Authorization `code` is issued to the authenticated Wallet Instance (**RFC 6749**) and was not replied.
2. It MUST ensure the Authorization `code` is valid and has not been previously used (**RFC 6749**).

3. It MUST ensure the `redirect_uri` matches the value included in the previous Request Object (see Section 3.1.3.1. of [OIDC]).
4. It MUST validate the DPoP Proof JWT, according to (**RFC 9449**) Section 4.3.
5. It MUST verify the `code_verifier` parameter according to the PKCE mechanism, ensuring it matches the `code_challenge` associated with the authorization code as defined in Section 4.6 of **RFC 7636**; otherwise, the request MUST be rejected (*CI_061a* (page 776)).

```
POST /token HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwieCI6IjR2dDhNdEFISmlsM
zBDNnpUTmt2c0VVcnlHTEUtQW5BNkc5LV8xa3l5Rk0iLCJ5IjoiTWdiNTFfbjNSRjNtbHNtS3dMd0xtRUFqVmlJM3Q1bTVW
NTI2MFA5MzR3RSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIjoiR0VUIiwiaHR1Ij
oiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0ZWRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.3Tp1Z
lZ05PQYeZUHhiZwaQ1etqnwYwoiJHFR_JHb32381lMJL-8o2rE3VZ8X3yuqrGFfCVeP90Ln4J5r8ASIBg
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sImNsaWVudF9zdGF0dXMiOnsic
3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTMzNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3Rh
dHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH0sIndhbGxldF9uYW1lIjoiV2FsbGV0X3YxIiwid2FsbGV0X2xpbms
iOiJodHRwczovL2V4YW1wbGUuY29tL2dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2FsbGV0X3ZlcnNpb24iOiIxLjAuMC
IsImV4cCI6MTc0MDE1ODE2N30.j2M6GTp3L7SzDHO6KUDo7PCuqCzO0TsHDYwOezsrAuDres4F8hAhK2SymL6nTMHuInaAQ
M74QlMvfBD2bY9E-w
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
&redirect_uri=https://start.wallet.example.org/cb
```

Step 11 (Token Response): The Credential Issuer validates the request. If successful, the Issuer provides the Wallet Instance with an Access Token and, optionally, a Refresh Token, both bound to the DPoP key.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
```

```
{
    "access_token": "eyJ0eXAiOiJhdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwYT
UwMDk2YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsInN1YiI6ImQ0Z
TBiYjM4N2FhMjU1NmZmMzA2OTI1ZmRmYjlhNzY1IiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV0LmlwenMu
aXQvY3JlZGVudGlhbCIsImlhdCI6MTczOTk1Mjk0OCwiZXhwIjoxNzM5OTUzNTQ4LCJqdGkiOiJmOTY1NWNlYi1jNjVjLTQ
wMjUtOTM3OC1iNjY3MmI2MTQ5YmciLCJjbGllbnRfaWQiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsIm
NuZiI6eyJqa3QiOiI5NTE1NzRhZWUxYmI3OTA3YWUxZWMzMTA5ZGIyYjIyNSJ9fQ.XQ3TAo69f2iPNJuMYAS5HdOSN8G5wm
IRMcSR6ocLEHdDgTUNI0TrMk-KapG4fvu5_bdZkjWBJvbjXeuP-Jngpg",
    "refresh_token": "eyJ0eXAiOiJydCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwY
TUwMDk2YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsImNsaWVudF9p
ZCI6IjQ3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV
0LmlwenMuaXQiLCJpYXQiOjE3Mzk5NTI5NDgsIm5iZiI6MTczOTk1MzU0OCwiZXhwIjoxNzQyMzcyNzQ4LCJhdGgiOiJmVU
h5TzJyMlozRFo1M0VzTnJXQmIweFdYb2FOeTU5SWlLQ0Fxa3NtUUVvIiwianRpIjoiYzY5NTVjZWItYzY1Zi00MDI1LTkzN
zgtYjY2NzJiNjE0NWNmIiwiY25mIjp7ImprdCI6Ijk1MTU3NGFlZTFiYjc5MDdhZTFlYzMxMDlkYjJiMjI1In19.qiGM6E-
7zci2-3Nnk4OMD7Tv_leUcRPsFsqaBHDHxEEzsGXLNh9qDbLIBk9sujZGVT9xs-28jZhwD6VT-MGTGw",
    "token_type": "DPoP",
    "expires_in": 3600,
    "authorization_details": [
        {
            "type": "openid_credential",
            "credential_configuration_id": "dc_sd_jwt_Education_degree",
            "credential_identifiers": [ "education_degree_unibo_2017_l31_informatica", "educati
on_degree_unibo_2019_lm18_informatica", "education_degree_unibo_2022_phd_informatica" ]
        }
    ]
}
```


A non-normative example of the DPoP Access Token is given below.


```
{
    "typ": "at+jwt",
    "alg": "ES256",
    "kid": "c950c0e6fdeb5de50a50096b247af03c"
}
```

```
{
    "iss": "https://eaa-provider.example.org",
    "sub": "d4e0bb387aa2556ff306925fdfb9a765",
    "aud": "https://eaa-provider.example.org",
    "iat": 1739952948,
    "exp": 1739953548,
    "jti": "f9655ceb-c65c-4025-9378-b6672b6149bg",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "cnf": {
        "jkt": "951574aee1bb7907ae1ec3109db2b225"
    }
}
```

A non-normative example of the DPoP Refresh Token is given below.

```
{
    "typ": "rt+jwt",
    "alg": "ES256",
    "kid": "c950c0e6fdeb5de50a50096b247af03c"
}
```

```
{
    "iss": "https://eaa-provider.example.org",
    "sub": "d4e0bb387aa2556ff306925fdfb9a765",
    "aud": "https://eaa-provider.example.org",
    "iat": 1739952948,
    "exp": 1742372748,
    "nbf": 1739953548,
    "ath": "fUHyO2r2Z3DZ53EsNrWBb0xWXoaNy59IiKCAqksmQEo",
    "jti": "c6955ceb-c65f-4025-9378-b6672b6145cf",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "cnf": {
        "jkt": "951574aee1bb7907ae1ec3109db2b225"
    }
}
```

Step 12 (Nonce Request): According to Section 7.1 of [OpenID4VCI], the Wallet Instance sends an HTTP POST request to the Nonce Endpoint to obtain a fresh `c_nonce` that can be used to create the proof of possession of key material for the subsequent request to the Credential Endpoint (*WP_056a* (page 746)).

Below is a non-normative example of a Nonce Request:

```
POST /nonce HTTP/1.1
Host: eaa-provider.example.org
Content-Length: 0
```

Step 13 (Nonce Response): The Credential Issuer provides the `c_nonce` to the Wallet Instance. The parameter `c_nonce` is a string value, which MUST be unpredictable and is used later by the Wallet Instance in Step 16 to create the proof(s) of possession of the key (`proofs` claim) and it is the primary countermeasure against key proof replay attack. Note that, the received `c_nonce` value can be used to create the proof(s) as long as the Issuer provides the Wallet Instance with a new `c_nonce` value.

Below is a non-normative example of a Nonce Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
```

```
{
    "c_nonce": "ts_EtUQs0ieiIS1NYNBHEQSoy3ct4gpy-4FZKwHilkY"
}
```

Step 14 (DPoP Proof for Credential Endpoint): The Wallet Instance creates a DPoP Proof JWT using the same key in **Step 8** and according to (**RFC 9449**) Section 4 (*WP_056b* (page 746)).

Note

If the Wallet Instance requests a batch issuance of Digital Credentials, the flow continues with **Step 17**. Otherwise Steps **15-16** are performed.

Step 15 (Proof of Possession of Credential): The Wallet Instance for requesting the Digital Credential creates a proof of possession with `c_nonce` obtained in **Step 13** using the private key used for the DPoP. The `jwk` value in the proof parameter MUST be equal to the public key referenced in the DPoP (*WP_056c* (page 746)).

Step 16 (Credential Request): The Wallet Instance sends a request for the Digital Credential to the Credential endpoint. This request MUST include the Access Token, DPoP Proof JWT, Credential type, proof (which demonstrates possession of the key) as per *WP_056* (page 746). The `proofs` parameter MUST be an object that contains evidence of possession of the cryptographic key material to which the issued Digital Credential will be bound. To verify the proof, the Credential Issuer conducts the following checks at the Credential endpoint:

1. The JWT proof MUST include all required claims as specified in the table of Section credential-issuance-endpoint:Token Request.
2. The key proof MUST be explicitly typed using header parameters as defined for the respective proof type.
3. The header parameter `alg` MUST indicate a registered asymmetric digital signature algorithm, and MUST NOT be set to *none*.
4. The signature on the key proof MUST be verified using the public key specified in the header parameter.
5. The header parameter MUST NOT contain a private key.
6. The signature on the Key Attestation JWT as the value of the `key_attestation` header parameter MUST be verified using the Wallet Provider's public key which is identified by the `kid` header parameter inside the Key Attestation JWT.
7. If a `c_nonce` value was previously provided by the server, the `nonce` claim in the JWT MUST match this `c_nonce` value. Furthermore, the creation time of the JWT, as indicated by the `iat` claim or a server-managed timestamp via the `nonce` claim, MUST be within an acceptable window of time as determined by the server.

Note

The Credential Issuer MUST register all the issued Credentials for their later revocation, if needed.

Note

It is RECOMMENDED that the public key contained in the `jwt_proof` be specifically generated for the requested Credential (fresh cryptographic key) to ensure that different issued Credentials do not share the same public key, thereby remaining unlinkable to each other.

A non-normative example of the Credential Request is provided below.

```
POST /credential HTTP/1.1
  Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA

{
    "credential_identifier": "education_degree_unibo_2017_l31_informatica",
    "proofs": {
        "jwt": [
```

```
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESWdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TlRRME1U
Y3lNVEkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkZlEuWndqZUpUQTNnc3Z0Mk1NdzlTMy1iZzREblY1TGRRRXB6OUYxcmVCRWlDblRuaURVa
E9jS010RzVnNmZwRFBVMWRQMjBzdTlXVXR1Rld6c0d1b0dhNFEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.2Z
QF2kYONuzkqIJ7nJxSl41-f0bTVSBdDpLFSrhA6PfmrcQJzBlTsv7FuGtXN-6mF3tZroklA2zjLmcaSCP3QQ"
        ]
    }
}
```


Where a non-normative example of the decoded content of the `jwt` parameter is represented below, without encoding and signature.


```
{
    "typ": "openid4vci-proof+jwt",
    "alg": "ES256",
    "jwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "4Fi3wDIgfbeQK624NyR7XOvAk3shaWy935yv5SHBqMw",
        "y": "D-R10rPxM0yqTJF6I3v-73USCU_6FHazyIJPLEsyUFE"
    },
    "key_attestation": "eyJhbGciOiJFUzI1NiIsImtpZCI6IjV0NVlZcEJoTi1FZ0lFRUk1aVV6cjZyME1SMDJMblZ
RME9tZWttTktjalkiLCJ0cnVzdF9jaGFpbiI6WyJleUpoYkdjaU9pSkZVei4uLjZTMEEiLCJleUpoYkdjaU9pSkZVei4uLm
pKTEEiLCJleUpoYkdjaU9pSkZVei4uLkg5Z3ciXSwidHlwIjoia2V5YXR0ZXN0YXRpb24rand0In0.eyJhdHRlc3RlZF9rZ
XlzIjpbeyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6IjRGaTN3RElnZmJlUUs2MjROeVI3WE92QWszc2hhV3k5MzV5
djVTSEJxTXciLCJ5IjoiRC1SMTByUHhNMHlxVEpGNkkzdi03M1VTQ1VfNkZIYXp5SUpQTEVzeVVGRSJ9XSwia2V5X3N0b3J
hZ2Vfc3RhdHVzIjp7InN0YXR1cyI6eyJzdGF0dXNfbGlzdCI6eyJpZHgiOjQxMiwidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW
9uX3VybC9zdGF0dXNsaXN0cy8xIn19LCJleHAiOjE1NDQxNzIxMjR9LCJpYXQiOjE2ODcyODExOTUsImV4cCI6MTY4NzI4O
DM5NSwia2V5X3N0b3JhZ2UiOlsiaXNvXzE4MDQ1X21vZGVyYXRlIl0sInVzZXJfYXV0aGVudGljYXRpb24iOlsiaXNvXzE4
MDQ1X21vZGVyYXRlIl19.LG9hkaoGmznUAL43gytXyEOoH7J-kKZQj_sG7Row2DUl0OYArV3mJnDmIuK_4-YVPg4AM03xV0
28BvFRwv5XDw"
}

{
    "iss": "47b982369791d08003a7283f059cb0d1",
```

```
    "aud": "https://eaa-provider.example.org",
    "iat": 1705570055,
    "nonce": "ts_EtUQs0ieiIS1NYNBHEQSoy3ct4gpy-4FZKwHilkY"
}
```

Step 17 (Fresh Digital Credential Keys Generation): The Wallet Instance generates N fresh Credential key pairs, where the number of key pairs (N) is determined by the value defined in the `batch_size` (*WP_058a* (page 746)).

Step 18 (Proofs of Possession of Credentials): The Wallet Instance MUST generate N key proofs using provided `c_nonce` in **Step 13** and one for each Credential in the batch. The number of key proofs (N) is defined by the `batch_size` value (*WP_058b* (page 746)).

Note

The `c_nonce` value in all the jwt proofs is identical and it is not needed to obtain separate nonce values per proof.

Step 19 (Batch Credential Request): The Wallet Instance sends a request for the batch of Digital Credential to the Credential endpoint. This request MUST include the Access Token, DPoP Proof JWT, Credential type, proofs (which demonstrates possession of the keys). The proofs parameter MUST set using a JSON object containing two or more evidence of possession of the cryptographic key materials to which the issued batch of Digital Credential will be bound. To verify the proofs, the Credential Issuer, in addition to the predefined checks in **Step 16**, must ensure the `jwk` attribute in each key proofs is unique (*WP_058* (page 746)).

```
POST /credential HTTP/1.1
  Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA

{
    "credential_identifier": "education_degree_unibo_2017_l31_informatica",
    "proofs": {
        "jwt": [
```

```
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESWdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TlRRME1U
Y3lNVEkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE9EQTBOVjl0YjJS
bGNtRjBaU0pkZlEuWndqZUpUQTNnc3Z0Mk1NdzlTMy1iZzREblY1TGRRRXB6OUYxcmVCRWlDblRuaURVaE9jS010RzVnNmZwRFBVMWRQMjBzdTlXVXR1Rld6c0d1b0dhNFEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.2Z
QF2kYONuzkqIJ7nJxSl41-f0bTVSBdDpLFSrhA6PfmrcQJzBlTsv7FuGtXN-6mF3tZroklA2zjLmcaSCP3QQ",
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESWdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TlRRME1U
Y3lNVEkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE9EQTBOVjl0YjJS
bGNtRjBaU0pkZlEuUGRFYWhhVlAwT2dqOTRwYTdiSGcxY1BKVmF6bHRibGJORHNxU2R1QVNDUkxjY1Ayc
GxCaDYwNHZNcWp5WHpSV3pDN0xCbXZpQ29IN1hjVEJReXJKaUEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.mT
hywjS7lYdmYpR3FY0cVVQJlqaiOPfnEmQU2UcNt4mP6Gl6qgAyhuwqAZoAx5yEeZTzRVzwt44NMrk6mmra8w"
        ]
    }
}
```


The decoded content of `jwt` elements in the `jwt` array is similar to what is explained in **Step 16**.

Steps 20-24 (Credential Response): The Credential Issuer MUST validate the *DPoP JWT Proof* based on the steps defined in Section 4.3 of (**RFC 9449**) and whether the *Access Token* is valid and suitable for the requested Credential. The Credential Issuer MUST validate all the key proofs that are provided within `proofs` (**Step 15** and **Step 18**) parameter that the

new Credentials MUST be bound to, according to OpenID4VCI Appendix F1. If all checks succeed, the Credential Issuer returns the issued Credential inside the `credentials` parameter. The number of elements in the Credentials array matches the number of the keys that the Wallet Instance has provided either via the `proofs` parameter (**Step 16** and **Step 19**). The Wallet Instance MUST perform the following checks before proceeding with the secure storage of the Credential(s):

1. It MUST check that the PID/(Q)EAA contained in the Credential Response contains all the mandatory parameters and values are validated according to *Table of the Credential response parameters* (page 586) (*WP_059* (page 746)).
2. It MUST check the Credential integrity by verifying the signature using the algorithm specified in the `alg` header parameter of SD-JWT (*Digital Credential Data Model* (page 357)) and the public key that is identified using the `kid` header of the SD-JWT (*WP_062a* (page 746)).
3. It MUST check that the received Digital Credential(s) (in the `credential` claim) matches the requested Credential type and complies with the specific schema of that Credential defined in *Digital Credential Data Model* (page 357) (*WP_060* (page 746)).
4. It MUST process and verify the Credential in SD-JWT VC format (according to SD-JWT Section 4.) or mdoc-CBOR format (*WP_062* (page 746)).
5. It MUST verify the Trust Chain in the header of SD-JWT VC to verify that the Credential Issuer is trusted (*WP_061* (page 746)).

If the checks above are successful, the Wallet Instance requests the User's consent to store the Digital Credential(s). Upon receiving consent, the Wallet Instance securely stores the Digital Credential(s) (*WP_063* (page 746)).

Below is a non-normative example of a successful response containing a Credential in SD-JWT VC format.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
```

```
            "credential": "eyJ0eXAiOiJ2YytzZC1qd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRl
YjVkZTUwYTUwMDk2YjI0N2FmMDNjIn0.eyJfc2QiOlsiQ1JJQkdpbWhhbE1TSzhLZUxmNzg2N0w4cHV6MEZnRTVaS1VXLW1
2N0hiYyIsIlhhZjVJZFVGc0UzYWtabEszT0E5d3dHcjJKcVAwUU01M3BBY2hiempRZmMiLCJjR2FuQVdySG9WQVoyalBhNX
k0SzE3U0xpYWFKcGRNUF9PdnBmTGx0VWJjIiwiNEFuZU1ZVVAxRWh3emRHdkRQOEhobnRaRGN1ejZrOHhHWVJ0NXo0SHh0S
SIsIjRuYTVXSHRMYzdrYnNxdHFVaHd6WXdVdUQtY3hKVmdENmRaLTl0ZUdhZ3MiLCJiZS10a2U3YVU0WmhDNWUxcGZqRjcx
UWpFRzRsZG1IaFRoUFl1TnQyOHo0IiwiSnNkaHhTMWRVTTJaN29MUmZYWVJvOFFEaDJ4M1dQNkdILUJnY21DdzJGayIsIlV
obXBRSy1HS3hzaHJwXzZwYVpfZzROVG5fX29aeVdOb01zTGNaMUhlMjgiLCJOdDdQU3RxMkEyWkliUHBHdTJpdmVSek9rbW
pYUEN1V0RBdy0tdktSQTBjIiwiV3Rtck5JVzFTVkN0UnZNUF9UM2YtRGlhRUdHNS1iVk5rWGJRckowRnpzRSJdLCJleHAiO
jE3NDY1MTU5NzIsImlzcyI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmlwenMuaXQiLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25v
V1hGWkFmSGt4WnNSR0M5WHMiLCJzdGF0dXMiOnsic3RhdHVzX2F0dGVzdGF0aW9uIjp7ImNyZWRlbnRpYWxfaGFzaF9hbGc
iOiJzaGEtMjU2In19LCJ2Y3QiOiJFZHVjYXRpb25fZGVncmVlIiwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOnsiandrIj
p7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoicFZVM2phdHU0YTN0azljOWFvd1ZnTHlCQl9ySjdNLTNXbGprMWVqV
XoyRSIsInkiOiJUTDVPTnZSLUlnYXJuZ3J6NWpkdnNwb2ZmekZ3Y2pQUnRGVWtlbmVIRUkwIn19fQ.v9ynFXhKXPOhQSMmu
LvIBKRWfPEPDf4QwDoNmDOjMROxr5J4Hshh9mBEM5qohH_PDE62i1TLc36C65jFYa7x3A~WyJjQ0ZDeXljV1J4alZINkZUR
VR5OTd3IiwidGF4X2lkX2NvZGUiLCJSU1NNUkE4MFIwMUg1MDFCIl0~...~WyIwQUx5SzRfUi1aVUpTekVKdW5HTFdRIiwi
aWF0IiwiMTc0NzExOTU5NSJd"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
}
```


Below is a non-normative example of a successful response containing a Credential in mdoc format.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
            "credential": "omppc3N1ZXJBdXRohEOhASahG...ArQwggKwMIICVqADAgEC"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
}
```

Below is a non-normative example of a successful response containing a batch of Credentials in SD-JWT VC format:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
```

```
            "credential": "eyJ0eXAiOiJ2YytzZC1qd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRl
YjVkZTUwYTUwMDk2YjI0N2FmMDNjIn0.eyJfc2QiOlsiQ1JJQkdpbWhhbE1TSzhLZUxmNzg2N0w4cHV6MEZnRTVaS1VXLW1
2N0hiYyIsIlhhZjVJZFVGc0UzYWtabEszT0E5d3dHcjJKcVAwUU01M3BBY2hiempRZmMiLCJjR2FuQVdySG9WQVoyalBhNX
k0SzE3U0xpYWFKcGRNUF9PdnBmTGx0VWJjIiwiNEFuZU1ZVVAxRWh3emRHdkRQOEhobnRaRGN1ejZrOHhHWVJ0NXo0SHh0S
SIsIjRuYTVXSHRMYzdrYnNxdHFVaHd6WXdVdUQtY3hKVmdENmRaLTl0ZUdhZ3MiLCJiZS10a2U3YVU0WmhDNWUxcGZqRjcx
UWpFRzRsZG1IaFRoUFl1TnQyOHo0IiwiSnNkaHhTMWRVTTJaN29MUmZYWVJvOFFEaDJ4M1dQNkdILUJnY21DdzJGayIsIlV
obXBRSy1HS3hzaHJwXzZwYVpfZzROVG5fX29aeVdOb01zTGNaMUhlMjgiLCJOdDdQU3RxMkEyWkliUHBHdTJpdmVSek9rbW
pYUEN1V0RBdy0tdktSQTBjIiwiV3Rtck5JVzFTVkN0UnZNUF9UM2YtRGlhRUdHNS1iVk5rWGJRckowRnpzRSJdLCJleHAiO
jE3NDY1MTU5NzIsImlzcyI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmlwenMuaXQiLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25v
V1hGWkFmSGt4WnNSR0M5WHMiLCJzdGF0dXMiOnsic3RhdHVzX2F0dGVzdGF0aW9uIjp7ImNyZWRlbnRpYWxfaGFzaF9hbGc
iOiJzaGEtMjU2In19LCJ2Y3QiOiJFZHVjYXRpb25fZGVncmVlIiwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOnsiandrIj
p7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoicFZVM2phdHU0YTN0azljOWFvd1ZnTHlCQl9ySjdNLTNXbGprMWVqV
XoyRSIsInkiOiJUTDVPTnZSLUlnYXJuZ3J6NWpkdnNwb2ZmekZ3Y2pQUnRGVWtlbmVIRUkwIn19fQ.v9ynFXhKXPOhQSMmu
LvIBKRWfPEPDf4QwDoNmDOjMROxr5J4Hshh9mBEM5qohH_PDE62i1TLc36C65jFYa7x3A~WyJjQ0ZDeXljV1J4alZINkZUR
VR5OTd3IiwidGF4X2lkX2NvZGUiLCJSU1NNUkE4MFIwMUg1MDFCIl0~...~WyIwQUx5SzRfUi1aVUpTekVKdW5HTFdRIiwi
aWF0IiwiMTc0NzExOTU5NSJd"
```


```
        },

        {
            "credential": "YXNkZnNhZGZkamZqZGFza23....29tZTIzMjMyMzIzMjMy"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
}
```

Note

When the Wallet Instance receives a new batch of the same Credential with the same claims, the Wallet MUST delete previous Credentials (*WP_073a* (page 746)).

Note

If the requested Credential cannot be issued immediately and requires more time, the Credential Issuer SHOULD support the Deferred Flow (step 27) as specified in Section credential-issuance-endpoint:Deferred Endpoint (*WP_065–067* (page 746)). Additionally, in the case of batch issuance, the same `transaction_id` retrieves all Credentials that are requested in the batch.

Step 25 (Notification Request): According to Section 11.1 of [OpenID4VCI], the Wallet sends an HTTP POST request to the Notification Endpoint using the *application/json* media type as in the following non-normative example (*WP_064* (page 746)).

```
POST /notification HTTP/1.1
Host: eaa-provider.example.org
```

```
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA

{
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942",
    "event": "credential_accepted"
}
```

Step 26 (Notification Response): When the Credential Issuer has successfully received the Notification Request from the Wallet, it MUST respond with an HTTP status code *204* as recommended in Section 11.2 of [OpenID4VCI]. Below is a non-normative example of response to a successful Notification Request:

```
HTTP/1.1 204 No Content
```

Note

In some cases there are multiple sets of Credential data available in the Authentic Source system, and the User may be interested in obtaining more than one Credential. In these cases the Issuance Flow remains the same as described in *Low-Level Issuance Flow* (page 422). In the Token Response (**Step 11**), the Credential Issuer generates a unique identifier (`credential_identifier`) for each Credential Dataset provided in the *AttributeClaims* parameter of the e-service PDND *Get Attribute Claims* (page 603). Therefore, the `credential_identifiers` array within the `authorization_details` object contains the identifiers of all Credentials available for issuance. The Wallet sends a Credential Request (**Step 16**) for each identifier, obtaining multiple distinct Credentials that are individually shown to the User for the acceptance. Finally, the Credential Issuer is notified by the Wallet of the outcome through the credential-issuance-endpoint:Notification Endpoint (*WP_057* (page 746)).

Note

For batch-issued Digital Credentials, a single `notification_id` covers the entire batch-issued Credentials. The notification response (e.g. `credential_accepted` or `credential_stored`) applies to all Credentials, any partial failure is treated as a batch failure.

Refresh Token Flow

To use the Deferred, Credential Request, and Notification endpoints, the Wallet Instance MUST present a valid DPoP Access Token to the Credential Issuer. However, when these endpoints are used in the Deferred Flow, for re-issuing or notifying the deletion of a Digital Credential, the Access Token might expire, as it is designed to be short-lived and these actions MAY occur days later (*WP_066b* (page 746)). To address this, the specification RECOMMENDS the use of Refresh Tokens.

An Access Token obtained as a result of a Refresh Token flow MUST be limited to:

- the Deferred endpoint to obtain a new Digital Credential after time set in the parameter `interval` or when it is notified as ready to be issued (*WP_066* (page 746));
- the Notification endpoint, to notify the deletion of a Digital Credential to the Credential Issuer;
- the Credential endpoint, to refresh a Digital Credential that is already present in the Wallet Instance (also called Digital Credential re-issuance, see section *Re-Issuance Flow* (page 448)).

To mitigate the impact of a stolen Refresh Token, the Refresh Tokens MUST be DPoP. These aspects are detailed and discussed in Section *Security Considerations* (page 447).

Figure below shows how to obtain a new DPoP Access Token and a new DPoP Refresh Token to the Token Endpoint.

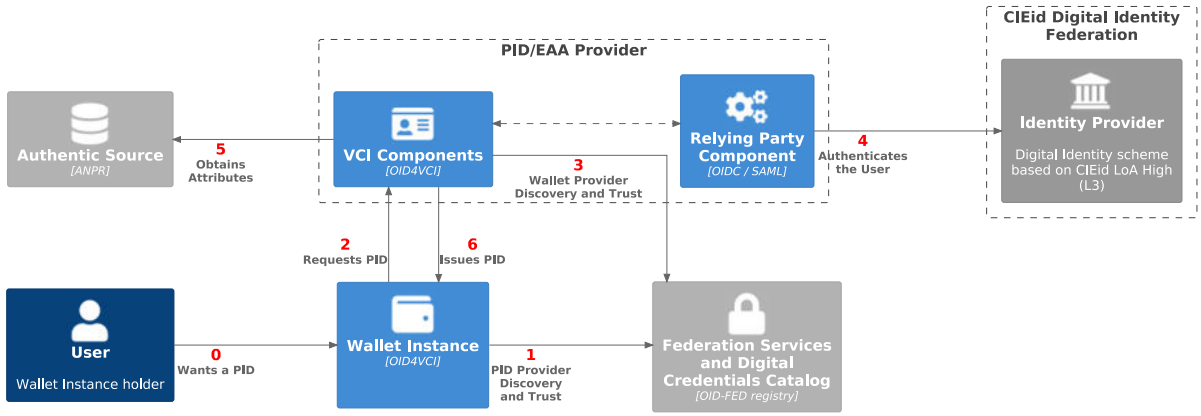


Fig. 12.6: Refresh Token Flow.

Note

The refresh of a Token may be triggered by different actions (e.g., User deletion of a Digital Credential). In each case, Wallet Instances are supposed to be running and the corresponding cryptographic material unlocked.

Step 1: The Wallet Instance MUST create a fresh DPoP Proof JWT and a fresh Wallet Instance Attestation proof of possession for the token request of the Credential Issuer (*WP_068a* (page 746)).

Step 2: To refresh a DPoP-bound Access Token, the Wallet Instance sends a token request using the parameter `grant_type` set to `refresh_token`, including the DPoP header and the OAuth Client Attestation headers (*WP_068* (page 746)). A non-normative example of the token request for a DPoP Access Token using a Refresh Token is shown below.

```
POST /token HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwieCI6IjR2dDhNdEFISmlsM
zBDNnpUTmt2c0VVcnlHTEUtQW5BNkc5LV8xa3l5Rk0iLCJ5IjoiTWdiNTFfbjNSRjNtbHNtS3dMd0xtRUFqVmlJM3Q1bTVW
NTI2MFA5MzR3RSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIjoiR0VUIiwiaHR1Ij
oiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0ZWRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.3Tp1Z
lZ05PQYeZUHhiZwaQ1etqnwYwoiJHFR_JHb32381lMJL-8o2rE3VZ8X3yuqrGFfCVeP90Ln4J5r8ASIBg
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sImNsaWVudF9zdGF0dXMiOnsic
3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTMzNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3Rh
dHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH0sIndhbGxldF9uYW1lIjoiV2FsbGV0X3YxIiwid2FsbGV0X2xpbms
iOiJodHRwczovL2V4YW1wbGUuY29tL3dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2FsbGV0X3ZlcnNpb24iOiIxLjAuMC
IsImV4cCI6MTc0MDE1ODE2N30.j2M6GTp3L7SzDHO6KUDo7PCuqCzO0TsHDYwOezsrAuDres4F8hAhK2SymL6nTMHuInaAQ
M74QlMvfBD2bY9E-w
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA

grant_type=refresh_token
&refresh_token=eyJ0eXAiOiJydCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwYTUwMDk2
YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsImNsaWVudF9pZCI6IjQ
3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV0Lmlwen
MuaXQiLCJpYXQiOjE3Mzk5NTI5NDgsIm5iZiI6MTczOTk1MzU0OCwiZXhwIjoxNzQyMzcyNzQ4LCJhdGgiOiJmVUh5TzJyM
lozRFo1M0VzTnJXQmIweFdYb2FOeTU5SWlLQ0Fxa3NtUUVvIiwianRpIjoiYzY5NTVjZWItYzY1Zi00MDI1LTkzNzgtYjY2
NzJiNjE0NWNmIiwiY25mIjp7ImprdCI6Ijk1MTU3NGFlZTFiYjc5MDdhZTFlYzMxMDlkYjJiMjI1In19.qiGM6E-7zci2-3
Nnk4OMD7Tv_leUcRPsFsqaBHDHxEEzsGXLNh9qDbLIBk9sujZGVT9xs-28jZhwD6VT-MGTGw
```

Step 3: The Credential Issuer validates the request according to the following checks:

- It MUST validate the OAuth-Client-Attestation-PoP parameter based on Section 5 of [OAUTH-ATTESTATION-CLIENT-AUTH].

- It MUST validate the DPoP Proof JWT, according to (**RFC 9449**) Section 4.3.
- It MUST check that the Refresh Token is not expired, not revoked and is bound to the same set of DPoP key as the ones used in the DPoP Proof JWT.

If the request checks are successful, the Credential Issuer generates a new Access Token and a new Refresh Token and these MUST be both bound to the DPoP key. Both the Access Token and the Refresh Token are then sent back to the Wallet Instance.

A non-normative example of a successful response is shown below.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "access_token": "eyJ0eXAiOiJhdCtqd3QiLCJhbGciOiJFU..",
    "refresh_token": "eyC3fiLdCtqd3QiLCJhbGciOiCL3..",
    "token_type": "DPoP",
    "expires_in": 3600
}
```

If the Refresh Token is expired or invalid, the Credential Issuer MUST issue an error, using the error type member set to `invalid_grant`. Therefore, to obtain the Digital Credential an issuance flow authenticating the User is required, as defined in Section *Low-Level Issuance Flow* (page 422).

Security Considerations

To mitigate the risks of Refresh Token compromise, the following protections are required:

- **Confidentiality** of Refresh Tokens MUST be guaranteed in transit and storage.
- **TLS-protected** connections MUST be used for token transmission.
- Refresh tokens MUST be **unguessable and secure from modification**.
- Authorization Servers MUST implement the following mechanism to **detect replay attacks**:
  - **Sender-Constrained Tokens**: Crypto-graphically bind the Refresh Token to the Wallet Instance according to **RFC 9449**. Access Tokens and Refresh Tokens MUST be bound to the same DPoP key. The DPoP Proof of the refresh token is required to refresh an Access Token. The same DPoP key MUST be used to generate Access Token DPoP Proofs in all the Credential Requests.
- **Limiting the use of Refresh Token**: As specified in OPENID4VC-HAIP: "Credential Issuers should be mindful of how long the usage of the refresh token is allowed to refresh

a Credential, as opposed to starting the issuance flow from the beginning. For example, if the User is trying to refresh a Credential more than a year after its original issuance, the usage of the refresh tokens is NOT RECOMMENDED." In this specification a new Digital Credential obtained performing the re-issuance flow SHOULD have the same expiration of the refreshed one. Thus, this specification does not allow for infinite refresh of Digital Credential with a Refresh Token. Once a Digital Credential expires, the User MUST complete the entire issuance process again, to obtain a new Digital Credential. This specification recommends to set a Refresh Token expiration duration, based on the sensitivity of the associated grant.

Note

Short-lived Wallet Instance Attestations and DPoP: Following the specification draft *OAuth 2.0 Attestation Based Client Authentication* (OAUTH-ATTESTATION-CLIENT-AUTH), the Authorization Server MUST bind the Refresh Token to the Client Instance. To prove this binding the Client Instance MUST use the Client Attestation mechanism when refreshing the Access Token and the Client Instance MUST use the same key that was presented in the `cnf.jwk` claim of the Client Attestation that was used when the Refresh Token was issued. However this requires that all issued Client Attestations MUST be bound to the same key, thus opening to unlinkability issues. In this specification, both OAUTH-ATTESTATION-CLIENT-AUTH and *OAuth 2.0 Demonstrating Proof of Possession (DPoP)* (**RFC 9449**) MUST be used. Using DPoP guarantees the binding of the Refresh Token with the Client Instance as stated in section 5 of **RFC 9449** *"the Refresh Token MUST be bound to the respective public key [...] a Client MUST present a DPoP proof for the same key that was used to obtain the Refresh Token each time that Refresh Token is used to obtain a new Access Token"* (*WP_068b* (page 746)). DPoP ensures that the Refresh Token is bound to the Wallet Instance.

Re-Issuance Flow

Re-issuance involves replacing Digital Credentials already stored in a Wallet Instance with new ones of the same document type. The new Digital Credentials MUST be issued by the same Credential Issuers that originally provided the existing ones to the same Wallet Instance.

To facilitate this, particularly in scenarios where User authentication is not strictly required, a Refresh Token (RT) flow MAY be used (see Section *Refresh Token Flow* (page 445) for more details). An Access Token obtained as a result of a Refresh Token flow MUST NOT be used to issue a Digital Credential that is not present in the Wallet Instance (first-time-issuance). The Refresh Token mechanism enables automated Credential replacement, streamlining the process for both the Credential Issuer and the User.

The re-issuance process outlined in this section is limited to the following scenarios:

- Data model/format technical update;
- User's attribute set update.

This flow does not apply to device migration or backup restore. In those cases the Wallet Instance MUST perform a Wallet-Initiated Authorization Code Issuance Flow as defined in Section *Low-Level Issuance Flow* (page 422) (see Section *Restore flow for Hardware Binding Credential* (page 304)).

In the first case, the new Digital Credential's Users attribute set will match the original one. For example, a Credential Issuer may need to update the Digital Credential metadata or data format without changing the User's attribute set. In this case, the direct involvement of the User is not mandatory for the replacement and storage of a Digital Credential.

In the second case, Credential Issuers may also need to modify one or more User's attribute values during re-issuance. In this case, the Wallet Instance MUST inform the User that the attribute data set has been changed and MUST then request the User's authorization to store the new Digital Credential.

In both cases, the newly issued Digital Credential MUST have the same expiry date as the previous one.

Re-issuance after Digital Credential expiration MUST always require User authentication.

The following diagram describes the Digital Credential re-issuance flow.

Step 1: The flow starts when the User opens the Wallet Instance: this step MAY be triggered either by a notification sent by the Credential Issuer (using e.g., one of the out-of-band communication contacts registered during the Issuance flow).

Step 2: The Wallet Instance MUST check the status of any stored Digital Credential, by retrieving either a valid Status List Token (following the flow described in Section *Status List Token* (page 413)) as per (*WP_069* (page 746)), if a valid one is not available. Then, the Wallet Instance MUST check (*WP_070* (page 746)) if any Digital Credential has status set to `0x03 - UPDATE` or `0x0F - ATTRIBUTE_UPDATE`.

If the conditions above are not met, the flow MUST be interrupted.

Otherwise, the Wallet Instance MUST check the related Access Tokens. If they are still valid, then Step 3 MAY be skipped (*WP_071* (page 746)).

Step 3: If the Access Token is expired and the Wallet Instance still has a valid Refresh Token, the Wallet Instance MUST obtain a new Access Token starting a Refresh Token Flow, according to Section *Refresh Token Flow* (page 445) (*WP_071a* (page 746)). The Refresh Token Flow enables the Wallet Instance to obtain a new Refresh Token and a new DPoP Access Token to refresh the Digital Credential. If the Refresh Token is expired, a new Issuance Flow authenticating the User is required (*WP_067* (page 746) and *WP_071b* (page 746)).

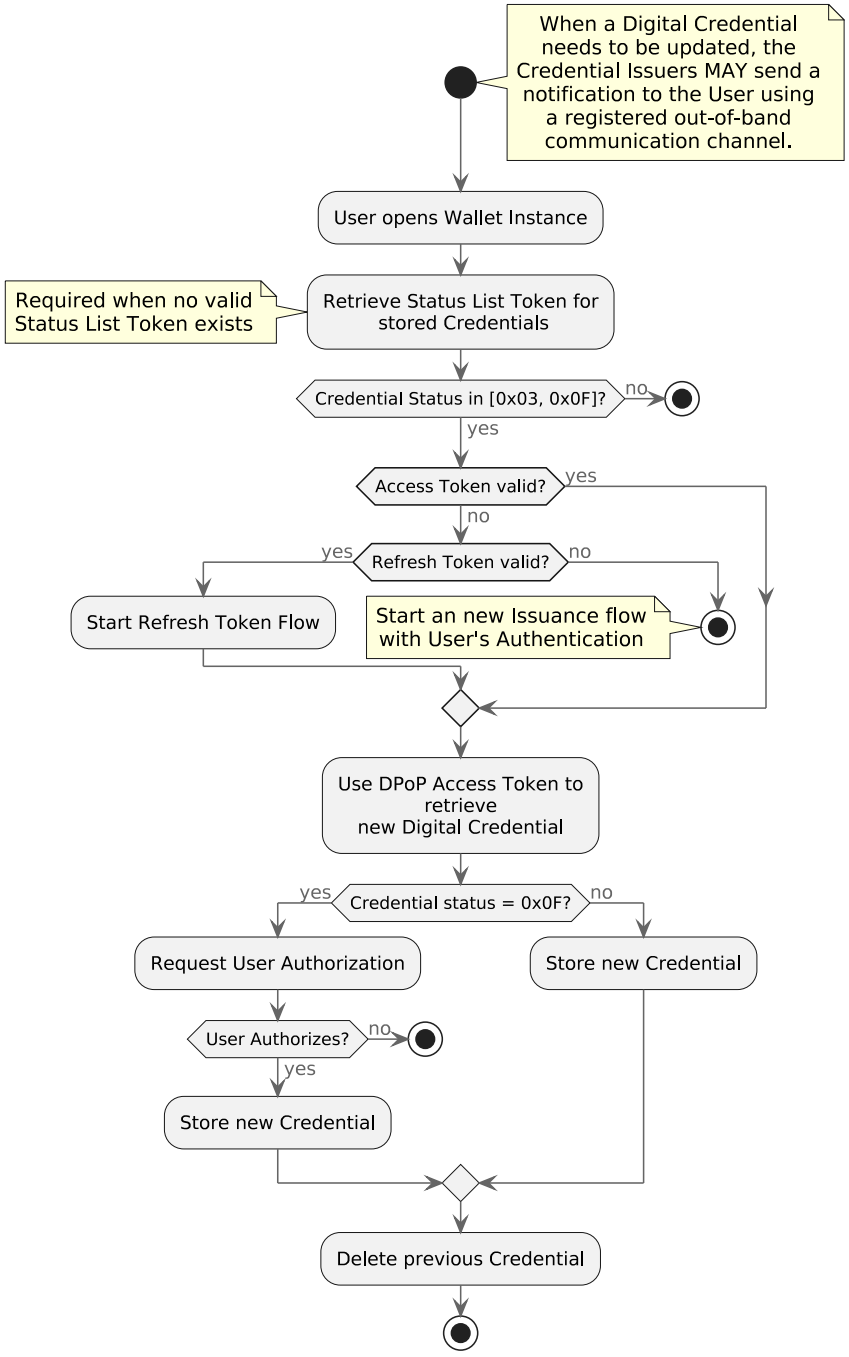


Fig. 12.7: Re-Issuance Flow Diagram.

Step 4: The Wallet Instance MUST use a valid DPoP Access Token to retrieve the new Digital Credential requesting it to the Credential endpoint following the steps from 12 to 22 of Figure 9 in Section *Low-Level Issuance Flow* (page 422) (*WP_072* (page 746)). When the new Digital Credential is successfully stored in the secure storage, the Wallet Instance MUST delete the previous one (*WP_073* (page 746)).

Note

Regardless of the Digital Credential revocation mechanism supported, if the Digital Credential status is set to `ATTRIBUTE_UPDATE`, the User's attribute set, in the refreshed Digital Credential, doesn't match the one in the stored Digital Credential. In this case, the Wallet Instance MUST request the User's authorization to store the new refreshed Digital Credential (*WP_074* (page 746)).

If instead, either the Digital Credential status is set to `UPDATE`, only the Credential metadata parameters have changed. In this case, the Wallet Instance SHOULD store the new Digital Credential without requiring explicit user authorization and consent (*WP_075* (page 746)).

Re-Issuance Flow: Security Considerations

To ensure the integrity and security of the re-issuance process, the following security considerations apply.

- Access Token limitations: An Access Token obtained as a result of a Refresh Token flow MUST NOT be used for the first-time issuance of a Digital Credential. This ensures that only existing Credentials in the Wallet Instance are updated.
- Credential expiry: The Credential Issuer MUST set the same expiry date for the re-issued Digital Credential as the previous one. This prevents indefinite Credential renewals without proper User authentication.
- User consent: For re-issuance processes triggered by attribute changes, User consent MUST be obtained before storing the new Digital Credential. This ensures that the User is aware of and agrees to the updated information.
- Sender-constrained Refresh Token: Refresh Tokens MUST be cryptographically bound to the Wallet Instance using DPoP protocol. This mitigates the risk of token misuse by ensuring that only the intended Wallet Instance (the same that originally has obtained the Digital Credential) can use that Refresh Token.

Note

During each credential issuance and, where supported, each re-issuance/refresh transaction executed through the flows described in this section, the Wallet Instance MUST create and maintain a corresponding transaction record in the Wallet Instance transaction log (see *Wallet Instance Dashboard and Transaction Logging* (page 299)).

The Wallet Instance MUST create a transaction record for each credential issuance process. The record MUST capture relevant contextual information necessary to ensure traceability and accountability of the transaction.

The record MUST be updated as the flow progresses to reflect the evolving transaction state and outcome context (e.g., the issued credential type(s) and quantity). In cases where the transaction does not complete successfully, the record MUST indicate the corresponding reason for non-completion.

Credential Offer Flow

A Credential Issuer or third party (e.g., Authentic Source, Registry, Catalogue) initiates Credential Issuance by sending a Credential Offer to the Wallet Instance.

To invoke the correct Wallet Instance, they need to know which Wallet Instance is installed on the User's device and the User wishes to use. This information SHOULD be obtained using the Selection Page described in *User Experience Design* (page 35).

- If the Selection Page is supported, the User selects the Wallet, and then the Credential Issuer or third party retrieves the Wallet metadata as described in *Wallet Metadata Retrieval Flow* (page 541). The Wallet Instance invocation mechanism depends on the `credential_offer_endpoint` parameter in the Wallet metadata:
  - If `credential_offer_endpoint` is available and contains an HTTPS URL (Universal Link), the Credential Issuer or third party SHOULD use that endpoint.
  - Otherwise, the Credential Issuer or third party MUST use one of the custom URL schemes: `openid-credential-offer://` (as defined in Section 4 of [OpenID4VCI]) or `haip-vci://` (as defined in Section 4.2 of [OPENID4VC-HAIP]). The Wallet Instance MUST support both custom URL schemes.
- In the case when the Credential Issuer or third party does not support the Selection Page, or the retrieval of the Wallet metadata may fail for some reason, the Credential Issuer or third party invokes the Wallet Instance using the custom URL scheme described above.

The Credential Offer can be transmitted by value or by reference:

- **By value** (`credential_offer` parameter): the Credential Offer object is directly embedded in the URI as a JSON-encoded string.

- **By reference** (`credential_offer_uri` parameter): the URI points to a resource hosted by the Credential Issuer or third party. The Wallet Instance retrieves the Credential Offer object by sending an HTTP GET request to that URI.

The Credential Offer object is a JSON object containing the parameters defined in Section 4.1.1 of [OpenID4VCI]. The table below specifies the parameters with IT-Wallet specific requirements:

Claim	Description	Reference
credential_issuer	It MUST be set with an HTTPS URL that uniquely identifies the Credential Issuer. The Wallet uses this parameter value to obtain the Credential Issuer's metadata.	Section 4.1.1 of [OpenID4VCI].
credential_configuration_ids	Array of Strings, each of them specifying a unique identifier of the Credential being described in the `credential_configurations_supported` map in the Credential Issuer Metadata (*WP_050b* (page 746)).	Section 4.1.1 of [OpenID4VCI].

continues on next page

Table 12.1 – continued from previous page

Claim	Description	Reference
grants	REQUIRED. It MUST contain `authorization_code` object with the following parameters: • **issuer_state**: OPTIONAL. Opaque string used to bind the subsequent Authorization Request with the Credential Issuer. It MAY be bound to a specific Credential Dataset provided by a certain Authentic Source. The Wallet MUST include it in the subsequent Authorization Request when present. It MUST be a URN and contain the following information: – *authenticSourceId:* REQUIRED. It MUST correspond to the `entity_id` value of the Authentic Source that provides the Credential Dataset(s) as registered in the *Authentic Source Registry* (page 163). – *datasetId:* REQUIRED. The unique identifier of the dataset provided by the Authentic Source as registered in the *Authentic Source Registry* (page 163). – *objectId:* OPTIONAL. Unique identifier of the Credential Dataset available from the Authentic Source. The `issuer_state` MUST follow the structure `urn:it-wallet:credential-offer:{authentic SourceId}:{datasetId}` when `objectId` is absent, or `urn:it-wallet:credential-offer:{authentic SourceId}:{datasetId}:{objectId}` when `objectId` is present. The optional `objectId` segment MUST be omitted when not available; an empty trailing segment MUST NOT be used. This URN value MUST be encrypted using the PDND public key related to the `GetAttributeClaims` e-service Consumer. • **authorization_server**: REQUIRED when the Credential Issuer uses more than one authorization server in its Issuer Solution. This string identifies the Authorization Server to use. The value MUST match with one of the values mapped in the `authorization_servers` array of the Credential Issuer metadata. It MUST NOT be used if `authorization_servers` is absent or it has no multiple entries.	Section 4.1.1 of [OpenID4VCI] and Section 4.1 of [OPENID4VC-HAIP].

continues on next page

Table 12.1 – continued from previous page

Claim	Description	Reference

Note

When using `credential_offer_uri` (by reference), the Credential Issuer or third party SHOULD use a unique URI for each Credential Offer or otherwise prevent caching of the URI, as recommended in Section 4.1.3 of [OpenID4VCI].

Non-normative Examples

Example 1: Credential Offer by value

The Credential Offer can be transmitted by value using any supported invocation method (custom URL scheme or Universal Link):

```
openid-credential-offer://?credential_offer=%7B%22credential_issuer%22%3A%22https%3A//credenti
al-issuer.example.org%22%2C%22credential_configuration_ids%22%3A%5B%22dc_sd_jwt_Education_degr
ee%22%5D%2C%22grants%22%3A%7B%22authorization_code%22%3A%7B%issuer_state%22%3A%22eyJhbGciOiJSU
0Et...F77QK8%22%7D%7D%7D
```

The decoded Credential Offer object:

```
{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["dc_sd_jwt_Education_degree"],
  "grants": {
    "authorization_code": {
      "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

Example 2: Credential Offer by reference from Credential Issuer

The QR Code or the href button contains:

```
openid-credential-offer://?credential_offer_uri=https%3A%2F%2Fcredential-issuer.example.org%2F
offers%2F8f3a2b1c
```

The Wallet Instance sends an HTTP GET request:

```
GET /offers/8f3a2b1c HTTP/1.1
Host: credential-issuer.example.org
Accept: application/json
```

The Credential Issuer responds:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["dc_sd_jwt_EuropeanDisabilityCard"],
  "grants": {
    "authorization_code": {
      "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

Example 3: Credential Offer by reference from third party

A third party, such as an Authentic Source, generates a Credential Offer for a Credential Issuer.

The QR Code contains:

```
openid-credential-offer://?credential_offer_uri=https%3A%2F%2Fauthentic-source.gov.example%2Fc
redential-offers%2Fabc123
```

The Wallet Instance sends:

```
GET /credential-offers/abc123 HTTP/1.1
Host: authentic-source.gov.example
Accept: application/json
```

The Authentic Source responds:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["mso_mdoc_mDL"],
  "grants": {
    "authorization_code": {
      "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

12.1.3 eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance

This Section defines an eID Substantial Authentication with MRTD Verification protocol that integrates within the IT-Wallet ID issuance flow as defined in the Section *Credential Issuance Low-Level Flows* (page 422) extending OAuth 2.0 flows to include:

- Electronic identification with Level of Assurance "Substantial" (LoA3).
- Electronic document verification as an additional identity verification layer.
- Session correlation and security binding between authentication steps.
- Integration with Credential issuance flows.

While CIEid with LoA High authentication remains the primary method for Wallet activation and IT-Wallet ID issuance, the eID Substantial Authentication with MRTD Verification mechanism defined in this Section provides an alternative approach to enhance service accessibility and usability, without compromising the overall security of the IT-Wallet ecosystem.

Note

This Section currently only supports the CIE id card for the MRTD verification protocol, the protocol described in this Section MAY be extended to support other MRTD Documents such as Electronic Passports.

Design Principles

The protocol adheres to the following design principles:

- **Standard Compatibility**: Extends existing authorization frameworks without breaking changes to the standard OAuth 2.0 Authorization Framework.
- **Multi-Step Authentication**: Implement progressive enforcement of authentication with MRTD verification.
- **Session Integrity**: Maintains secure session correlation across authentication steps.
- **Unified Flow**: Integrates multiple authentication factors in a single authorization session.

System Architecture

The system architecture comprises the following main components:

- **Wallet Instance:** It handles the IT-Wallet ID request according to IT-Wallet Specification, supporting additional cryptographic capabilities for MRTD/IAS Electronic Document reading according to ICAO 9303 and BSI-03110 specifications.
- **eID Substantial Authentication with MRTD Verification System:** Orchestrates the authentication flow, integrating LoA3 Identity Providers, Electronic Document Verification Service, and performing all the identity correlation checks to guarantee that the User requesting an IT-Wallet ID matches with the authenticated one.
  - **EAA Authorization Server:** Handles the authorization flow for IT-Wallet ID Issuance, coordinating the User authentication and the remote identity proofing.
  - **MRTD PoP Service:** Handles electronic document proof of possession with cryptographic document validation.
- **LoA3 Identity Provider:** Provides Electronic Identification Schemes with eIDAS LoA3 compliance (CIEid and SPID).
- **CIE National Registry:** Acts as the authoritative source for the CIE. Provides information related to the validity status of the document. It also optionally provides the MRZ data (document number, date of birth, expiry date, citizenship and gender) to improve the User experience.

High-Level Flow

The protocol consists of the following sequential phases:

1. **OAuth Authorization Request**: PAR and authorization request with eID Substantial Authentication with MRTD Verification.
2. **Primary Authentication**: LoA3 electronic identification (SPID/CIEid L2).
3. **MRTD Proof of Possession (PoP) Validation**: Electronic document reading and cryptographic verification (Proof of Possession, integrity, and authenticity checks).
4. **OAuth Authorization Response**: Final authorization code issuance.

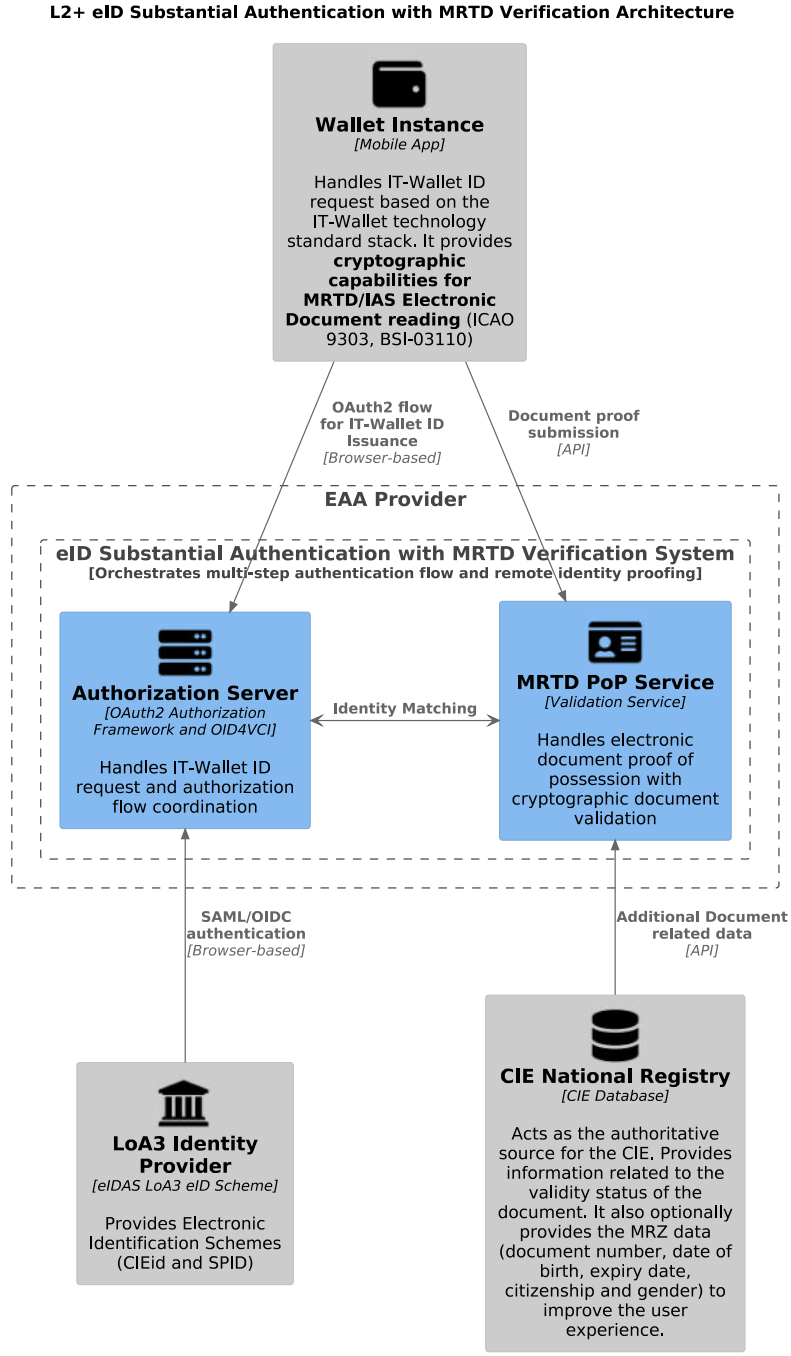


Fig. 12.8: eID Substantial Authentication with MRTD Verification System Architecture.

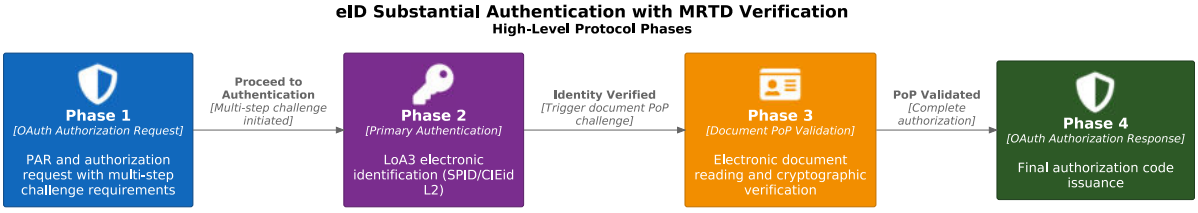


Fig. 12.9: eID Substantial Authentication with MRTD Verification High-Level Flow.

Session Management

The Authorization Server MUST maintain a unified session across all authentication steps, ensuring:

- **Session Continuity**: Single session identifier throughout the authentication flow.
- **State Correlation**: Authentication results from each step are correlated.
- **Security Binding**: Binding between authentication steps.

Since the EAA Authorization Server and the MRTD PoP Service operate within the EAA Provider boundary implementation, it is RECOMMENDED that the OAuth 2.0 session and the nonces used within the protocol flow are properly handled by both components seamlessly.

When both services operate within the same trust boundary, the following mechanisms are available for session correlation:

- Direct memory access to shared session stores.
- Internal service-to-service authentication using pre-shared keys.
- Synchronized nonce validation without external communication overhead.
- Unified audit logging and security event correlation.

When the EAA Authorization Server or MRTD PoP Service is deployed outside the EAA Provider boundary implementation, the *Security Considerations* (page 475) MUST be taken into account to harden the User authentication sessions management. These measures include but are not limited to secure session token handling, distributed session validation, and enhanced audit trail mechanisms.

Low-Level Flow

This section provides technical details about the IT-Wallet ID Issuance using Level of Assurance Substantial and remote identity proofing through verification of the electronic document using a MRTD Verification service.

Phase 1: OAuth Authorization Request

The Wallet Instance initiates the eID Substantial Authentication with MRTD Verification flow by submitting a Pushed Authorization Request (PAR), using a JWT-secured Authorization Request (JAR) and Rich Authorization Requests (RAR) as specified below.

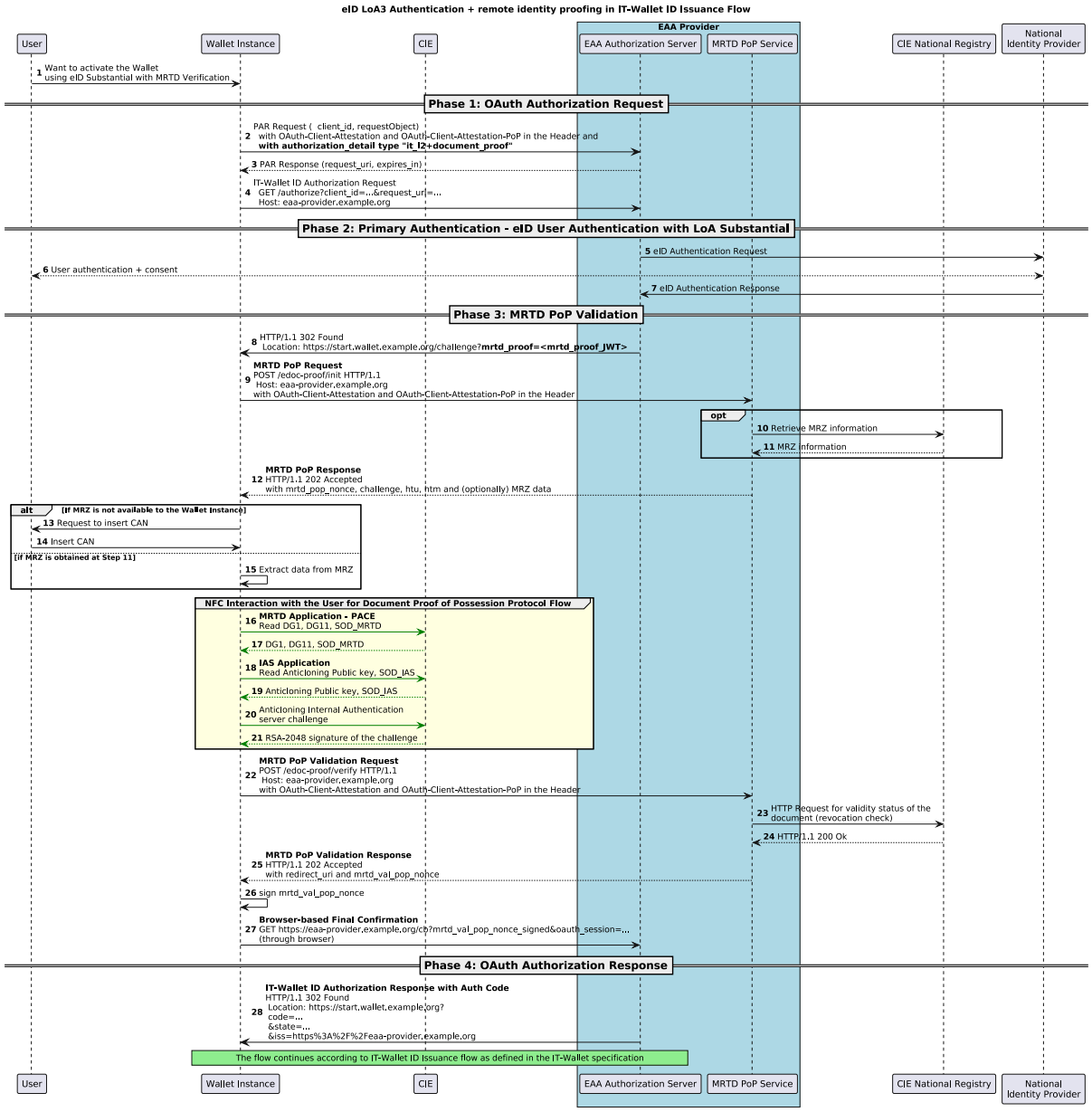


Fig. 12.10: eID Substantial Authentication with MRTD Verification Detailed Flow.

Authorization Details

The JWT Request Object MUST contain the same parameters as defined in credential-issuance-endpoint:Pushed Authorization Request Endpoint. The inclusion of the `it_l2+document_proof` Authorization Details Object and the authentication method applied by the Authorization Server are defined in credential-issuance-endpoint:User Authentication Method Selection (*WP_158* (page 739), *CI_199* (page 776)).

Below a non-normative example of PAR:

```
POST /as/par HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

client_id=47b982369791d08003a7283f059cb0d1&
request=eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWF1dGh6LXJlcStqd3QiLCJraWQiOiJiMDFiODIwOGQ5ZTZjYzg
zNGQ4N2RjMzU2YWI1MDE3MCJ9.eyJqdGkiOiJmODU1NWNlYi1jNjVjLTQwMjUtOTM3OC1iNjY3MmI2MTQ5YWYiLCJhdWQiO
iJodHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZyIsImlhdCI6MTcxNTg0MjU2MCwiZXhwIjoxNzE1ODQyODYwLCJy
ZXNwb25zZV90eXBlIjoiY29kZSIsInJlc3BvbnNlX21vZGUiOiJmb3JtX3Bvc3Quand0IiwiY2xpZW50X2lkIjoiNDdiOTg
yMzY5NzkxZDA4MDAzYTcyODNmMDU5Y2IwZDEiLCJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsIn
N0YXRlIjoiZnlaaU9MOUxmMkNlS3VOVDJKenhpTFJEaW5rMHVQY2QiLCJjb2RlX2NoYWxsZW5nZSI6IkU5TWVsaG9hMk93d
kZyRU1USmd1Q0hhb2VLMXQ4VVJXYnVHSlNzdFctY00iLCJjb2RlX2NoYWxsZW5nZV9tZXRob2QiOiJTMjU2Iiwic2NvcGUi
OiJlaWQiLCJhdXRob3JpemF0aW9uX2RldGFpbHMiOlt7InR5cGUiOiJvcGVuaWRfY3JlZGVudGlhbCIsImNyZWRlbnRpYWx
fY29uZmlndXJhdGlvbl9pZCI6ImRjX3NkX2p3dF9laWQifSx7InR5cGUiOiJpdF9sMitkb2N1bWVudF9wcm9vZiIsImNoYW
xsZW5nZV9tZXRob2QiOiJtcnRkK2lhcyIsImlkcGhpbnRpbmciOiJodHRwczovL2lkcC5leGFtcGxlLm9yZyIsImNoYWxsZ
W5nZV9yZWRpcmVjdF91cmkiOiJodHRwczovL3N0YXJ0LndhbGxldC5leGFtcGxlLm9yZy9jaGFsbGVuZ2UifV0sInJlZGly
ZWN0X3VyaSI6Imh0dHBzOi8vc3RhcnQud2FsbGV0LmV4YW1wbGUub3JnIn0.AuthRequestSign456_NoKidJWTSignatur
e-abc123def456ghi789jkl012mno345pqr678stu901vwx234yz567
```

The PAR Response and the Authorization Request are the same as in the IT-Wallet Specification.

Phase 2: Primary Authentication

Upon successful processing of the PAR, the Authorization Server redirects the User Agent to the Identity Provider for primary authentication, according to credential-issuance-endpoint:User Authentication Method Selection. The User completes the authentication flow and the Authorization Server correlates the authenticated identity with the active OAuth session.

In this flow **LoA3** (ISO/IEC 29115) maps to eIDAS **Substantial** and to the national L2 means (SPID L2 / CieID Substantial). **LoA High** maps to eIDAS High and CieID LoA High (CIE L3). A Substantial authentication MUST NOT be recorded or treated as High (*CI_205* (page 776)).

The EAA Authorization Server MUST ensure that the `mrtd_auth_session` parameter is maintained throughout this phase for proper session correlation with subsequent authentication steps.

After MRTD attributes become available (Phase 3), the EAA Provider MUST bind the identity verified by the IdP with the identity in the MRTD by comparing at least the tax identification number (*codice fiscale*) or an equivalent unique national identifier present in both sources. A mismatch MUST abort issuance (*WP_159* (page 739), *CI_200* (page 776)).

Note

If the User performs a LoA High authentication, Phase 3 MUST be skipped.

Phase 3: MRTD PoP Validation Flow

Following successful primary authentication, the Authorization Server initiates the electronic document proof of possession flow by providing the Wallet Instance with the necessary parameters for MRTD validation.

MRTD Proof JWT

The Authorization Server MUST provide a signed JWT containing the challenge requirements for document validation. The JWT structure is defined as follows:

Table 12.2: MRTD Proof JWT Header

Parameter	Type	Description
alg	string	REQUIRED. Signature algorithm.
typ	string	REQUIRED. It MUST be `mrtd-ias+jwt`.
kid	string	REQUIRED. Identifier of the EAA Provider's key that MUST be used to verify the signature of this JWT.

Table 12.3: MRTD Proof JWT Payload

Parameter	Type	Description
iss	string	REQUIRED. EAA Provider identifier.
aud	string	REQUIRED. Wallet Instance identifier.
iat	integer	REQUIRED. Issuance time (Unix timestamp).
exp	integer	REQUIRED. Expiration time (Unix timestamp).
status	string	REQUIRED. Challenge status. MUST be `require_interaction`.
type	string	REQUIRED. Type of the required interaction. It MUST be set to `mrtd+ias`.
mrtd_auth_session	string	REQUIRED. Session identifier.
state	string	REQUIRED. MUST be the same value as in the initial Request Object.
mrtd_pop_jwt_nonce	string	REQUIRED. Nonce value for replay protection. It MUST be obtained by the MRTD PoP Service that MUST have direct control over this value.
htu	string	REQUIRED. HTTP URI of the MRTD PoP initialization endpoint.
htm	string	REQUIRED. HTTP method for the MRTD PoP initialization request. MUST be `POST`.

A non-normative example is provided below:

```
HTTP/1.1 302 Found
Location: https://start.wallet.example.org/challenge?challenge_info=eyJhbGciOiJFUzI1NiIsInR5cCI
6Im1ydGQtaWFzK2p3dCIsImtpZCI6ImQ0ZjhlMDJlZWJhMDdhNTM3MmUxOGVjYzU4NzA5ZDA2In0.eyJpc3MiOiJodHRwcz
ovL3BpZC1wcm92aWRlci5leGFtcGxlLm9yZyIsImF1ZCI6IjQ3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwia
WF0IjoxNzUzNTU1MzU4LCJleHAiOjE3NTM1NTU2NTgsInN0YXR1cyI6InJlcXVpcmVfaW50ZXJhY3Rpb24iLCJ0eXBlIjoi
bXJ0ZCtpYXMiLCJtcnRkX2F1dGhfc2Vzc2lvbiI6Ind4cm9WckJZMk1DcTRkRE5HWEFDUyIsInN0YXRlIjoiZnlaaU9MOUx
mMkNlS3VOVDJKenhpTFJEaW5rMHVQY2QiLCJtcnRkX3BvcF9qd3Rfbm9uY2UiOiJub25jZTEiLCJodHUiOiJodHRwczovL2
Vkb2MtcHJvb2YvaW5pdCIsImh0bSI6IlBPU1QifQ.i6p_FN7qNNawyL4KnOV1r8FrNVjzd-7Ve1wEGASHNnlXwuJ1f216v0
Ml_KpVrq9yXkmOo_M2xZwih2SlHVfrzkuG3Pn7LWRL7dsyCtqEY2e58rFHjCa2miBnnKr0NU4wcBMMYe2_qKCOkA7SOa7us
NTBluBLMQ28GfiMbr3tcpfpM4rD0POKQcfijvNkNbh-VdOxM8GdHb6IQO_xfpsaSzd8cc0k5yIYCWjDTeINVKebIz4m9Rm2
JStvRrWUq8OCqkv-8dTJH9q-JXx0PzJC998RMwe6tqSL-kkE3dZLWwCJdP8Z7bITtowU49rEe-AkrGxVma4ANPq317umEfU
wmw
```

The Authorization Server MUST:

- Generate a unique challenge identifier with sufficient entropy (minimum 128 bits) for cryptographic security.
- Create MRTD Proof JWT with proper header (`alg`, `typ`, `kid`) and payload parameters (as defined in the table above).
- Sign the MRTD Proof JWT using its private key with the cryptographic algorithm chosen. See Section *Cryptographic Algorithms* (page 615).

- Construct a secure redirect URL with proper encoding to the Wallet Instance universal link.
- Set the redirect timeout to prevent indefinite waiting and handle timeout scenarios
- Optionally request MRZ information directly from the CIE National Registry using the authenticated User's taxpayer identification number.
- Maintain session correlation between the LoA High result and the document proof challenge

The Wallet Instance MUST:

- Validate the JWT signature using the EAA Provider's public key obtained through trust evaluation.
- Verify `aud` claim matches its `client_id`.
- Verify `iat` and `exp` claims indicate the token has a correct issuance date and has not expired.
- Verify that the `status` field is set to "require_interaction".
- Verify the authentication type matches the expected value `mrtd+ias`.
- Extract HTTP target URI (`htu`) and method (`htm`) for the next step.
- Handle JWT validation and network errors, and provide user feedback with appropriate retry mechanisms.
- Extract correlation parameters (`mrtd_auth_session`, `state`, `mrtd_pop_jwt_nonce`) for subsequent requests.

MRTD PoP Request

The Wallet Instance MUST send an HTTP POST Request to the MRTD PoP Service initialization endpoint with `application/json` as content type, including Wallet Attestation and Wallet Attestation JWT PoP in the header according to OAUTH-ATTESTATION-CLIENT-AUTH. The payload of the Request contains the following parameters:

Table 12.4: MRTD PoP Request Parameters

Parameter	Type	Description
mrtd_auth_session	string	REQUIRED. Session identifier for session binding.
mrtd_pop_jwt_nonce	string	REQUIRED. Nonce value obtained from the MRTD Proof JWT.

Below a non-normative example of an MRTD PoP Request:

```
POST /edoc-proof/init HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

{
  "mrtd_auth_session": "wxroVrBY2MCq4dDNGXACS",
  "mrtd_pop_jwt_nonce": "f42cccb7f1c8269f9d4aeefe339c6b13"
}
```

The Wallet Instance MUST:

- Validate the MRTD Proof JWT signature using the EAA Provider's public key.
- Verify the JWT `aud` claim matches its `client_id`.
- Ensure the JWT `exp` claim indicates the token has not expired.
- Extract the `mrtd_auth_session` and `mrtd_pop_jwt_nonce` values for correlation.
- Include valid Wallet Attestation according to OAUTH-ATTESTATION-CLIENT-AUTH.
- Handle network errors and implement appropriate retry mechanisms.

The MRTD PoP Service MUST:

- Validate the Wallet Attestation according to OAUTH-ATTESTATION-CLIENT-AUTH.
- Verify the `mrtd_auth_session` parameter matches an active session.
- Validate the `mrtd_pop_jwt_nonce` parameter corresponds to the one issued at the previous step.

The MRTD PoP Service MAY request the MRZ information (document number, date of birth, expiry date, citizenship and gender) directly to the CIE National Registry using the Tax payer's identification number of the authenticated User. In this case, the MRTD PoP Service is able to check if the authenticated User owns a valid CIE and If this is not the case, it MUST send an HTTP Error Response with HTTP Error code `access_denied`.

MRTD PoP Response

If the HTTP Request is successfully processed, the MRTD PoP Service MUST send a HTTP Response with code *202 Accepted* and `application/jwt` as content type. The JWT structure is defined as follows:

Table 12.5: MRTD PoP Response Parameters

Parameter	Type	Description
alg	string	REQUIRED. Signature algorithm.
typ	string	REQUIRED. It MUST be `mrtd-ias-pop+jwt`.
kid	string	REQUIRED. Identifier of the EAA Provider's key that MUST be used to verify the signature of this JWT.

Table 12.6: MRTD PoP Response Parameters

Parameter	Type	Description
iss	string	REQUIRED. EAA Provider identifier.
aud	string	REQUIRED. Wallet Instance identifier.
iat	integer	REQUIRED. Issuance time (Unix timestamp).
exp	integer	REQUIRED. Expiration time (Unix timestamp).
challenge	string	REQUIRED. Challenge data for cryptographic document validation. It SHOULD be the SHA-256 hash value of a random value with the `mrtd_auth_session` and timestamp for cryptographic binding.
mrtd_pop_ nonce	string	REQUIRED. Unique nonce value for the next step, preventing replay attacks.
mrz	string	OPTIONAL. Machine Readable Zone information including document number, date of birth, expiry date, citizenship and gender.
htu	string	REQUIRED. HTTP URI of the MRTD PoP validation endpoint.
htm	string	REQUIRED. HTTP method for the MRTD PoP validation request. MUST be `POST`.

Below a non-normative example of an MRTD PoP Response:

```
HTTP/1.1 202 Accepted
Content-Type: application/jwt; charset=utf-8

eyJhbGciOiJFUzI1NiIsInR5cCI6Im1ydGQtaWFzLXBvcCtqd3QiLCJraWQiOiJiM2YxYTZjMmU5ZDU0YThmOWMzZTdkMW
EyZjRiNmM3OCJ9.eyJpc3MiOiJodHRwczovL3BpZC1wcm92aWRlci5leGFtcGxlLm9yZyIsImF1ZCI6Imh0dHBzOi8vd2F
sbGV0LmV4YW1wbGUub3JnL2luc3RhbmNlLzEyMzQ1IiwiaWF0IjoxNzUzNTU1ODAwLCJleHAiOjE3NTM1NTYwMDAsIm1yd
GRfcG9wX25vbmNlIjoiOWYyYzRhN2UzYjFkOGM5YTZlNWY0YjJhMWMzZDdlOGYiLCJtcnoiOiIuLi4iLCJjaGFsbGVuZ2U
iOiIuLi4iLCJodHUiOiIuLi4iLCJodG0iOiIuLi4ifQ.signature
```

JWT decoded header:

```
{
  "alg":"ES256",
  "typ":"mrtd-ias-pop+jwt",
  "kid":"b3f1a6c2e9d54a8f9c3e7d1a2f4b6c78"
}
```

JWT decoded body:


```
{
  "iss":"https://eaa-provider.example.org",
  "aud":"https://wallet.example.org/instance/12345",
  "iat": 1753555800,
  "exp": 1753556000,
  "mrtd_pop_nonce":"9f2c4a7e3b1d8c9a6e5f4b2a1c3d7e8f",
  "mrz":"...",
  "challenge":"...",
  "htu":"...",
  "htm":"..."
}
```


The MRTD PoP Service MUST:

- Generate cryptographically secure challenge data with sufficient entropy (to be used in Anti-Cloning Internal Authentication protocol by the Wallet Instance), storing it with an appropriate expiration time. Moreover, the MRTD PoP Service MUST ensure challenge uniqueness to prevent reuse attacks.
- Create a new unique `mrtd_pop_nonce` for the next step to prevent replay attacks.
- Validate session continuity by ensuring the `mrtd_auth_session` parameter corresponds to an active session.
- Return HTTP *202 Accepted* status to indicate asynchronous processing initiation.
- Include proper Content-Type header (`application/jwt; charset=utf-8`).
- Handle service errors and return appropriate error responses.
- Extract and validate MRZ information if provided from external registry services.

The Wallet Instance MUST:

- Validate HTTP response status (*202 Accepted*) and content type.
- Parse JSON response and validate required parameters (`challenge`, `mrtd_pop_nonce`).
- Extract challenge data for cryptographic document validation.
- Store new `mrtd_pop_nonce` value securely for subsequent validation requests.
- Validate optional MRZ information if present in the response.
- Extract HTTP target URI (htu) and method (htm) for the next step.
- Handle errors, providing relative user feedback.
- Store challenge data temporarily.
- Prepare NFC reading session.

The Wallet Instance performs NFC-based electronic document reading and validation, then submits the evidence to the EAA Provider for final verification and identity correlation with the LoA3 authentication result.

MRTD PoP Validation Request

Upon all the evidence has been collected through NFC interaction with the Electronic Document, the Wallet Instance MUST send all data to the EAA Provider Authorization Server for the final validation and identity cross checks. The Wallet Instance MUST send a HTTP POST Request with `application/json` as content type, including Wallet Attestation and Wallet Attestation JWT PoP in the header according to OAUTH-ATTESTATION-CLIENT-AUTH. The payload of the Request contains the following parameters.

Table 12.7: MRTD PoP Validation Request Parameters

Parameter	Type	Description
mrtd_validation_jwt	string	REQUIRED. JWT containing document evidence, cryptographic proofs, Data Groups (DGs), and validation results.
mrtd_auth_session	string	REQUIRED. Session identifier for session binding.
mrtd_pop_nonce	string	REQUIRED. MUST match the value obtained from the MRTD PoP Response.

Validation JWT Structure

The Validation JWT (`mrtd_validation_jwt`) structure is given in the following table.

Table 12.8: Validation JWT Header Parameters

Header Parameter	Type	Description
alg	string	REQUIRED. Signature algorithm.
typ	string	REQUIRED. It MUST be `mrtd-ias+jwt`.
kid	string	REQUIRED. Identifier of the Wallet Instance's key that MUST be used to verify the signature of this JWT.

Table 12.9: Validation JWT Payload Parameters

Payload Parameter	Type	Description
iss	string	REQUIRED. Wallet Instance identifier.
aud	string	REQUIRED. EAA Provider identifier.
iat	integer	REQUIRED. Issuance time (Unix timestamp).
exp	integer	REQUIRED. Expiration time (Unix timestamp).
document_type	string	REQUIRED. Type of document validated. MUST be set to `cie`.
mrtd	JSON Object	REQUIRED. MRTD validation data containing Data Groups and SOD.
ias	JSON Object	REQUIRED. IAS validation data containing Anti-Cloning Public Key, and SOD.

MRTD Object Structure

The `mrtd` object contains the following fields:

Table 12.10: MRTD Object Structure

Field	Type	Description
dg1	string	REQUIRED. Base64-encoded Data Group 1 (MRZ information: document number, date of birth, expiry date, citizenship and gender).
dg11	string	REQUIRED. Base64-encoded Data Group 11 (additional personal data).
sod_mrtd	string	REQUIRED. Base64-encoded Security Object of Document for MRTD.

IAS Object Structure

The `ias` object contains the following fields:

Table 12.11: IAS Object Structure

Field	Type	Description
ias_pk	string	REQUIRED. Base64-encoded IAS public key in DER format.
sod_ias	string	REQUIRED. Base64-encoded Security Object of Document for IAS.
challenge_signed	string	REQUIRED. Base64-encoded signed challenge response.

Below a non-normative example of an MRTD PoP Validation Request:

```
POST /edoc-proof/verify HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

{
  "mrtd_validation_jwt":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3dhbGxldC5le
GFtcGxlLm9yZy9pbnN0YW5jZS8xMjM0NSIsImF1ZCI6Imh0dHBzOi8vcGlkLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0
IjoxNzUzNTU1NDAwLCJleHAiOjE3NTM1NTU3MDAsImRvY3VtZW50X3R5cGUiOiJjaWUiLCJtcnRkIjp7ImRnMSI6IlVEeEp
WRUU4VTAxSlZFZzhQRXBQU0U0OFBFcFBTRTRnVTAxSlZFZzhQREU1T0RBME1UVThUVDxQTnpjM056SXpNUT09IiwiZGcxMS
I6Ik1USXpORFUyTnpnNVFVSkRSRVZHUjBoSlNrdE1UVTVQVUVGT1IxSlRWRlZXV0ZsYVUwRkVSVVU9Iiwic29kX21ydGQiO
iJNSUlGempDQ0JMYWdBd0lCQWdJSVFPWTJLSkdGVFVJd0RRWUpLb1pJaHZjTkFRRUxCUUF3WHpFTE1Baz0ifSwiaWFzIjp7
Imlhc19wayI6Ik1JSUJJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBejEyMzQ1Njc4OTA9Iiwic29
kX2lhcyI6Ik1JSUZhRENDQkZDZ0F3SUJBZ0lKQUwyS0pHRlRVSXdEUVlKS29aSWh2Y05BUUVMQlFBd1h6RUxNQT09IiwiY2
hhbGxlbmdlX3NpZ25lZCI6ImExYjJjM2Q0ZTVmNjc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MGFiY2RlZjEyMzQ1Njc4OTBhY
mNkZWY9PSJ9fQ.xyz456abc789def012ghi345jkl678mno901pqr234stu567vwx890yz123",
  "mrtd_auth_session":"wxroVrBY2MCq4dDNGXACS",
  "mrtd_pop_nonce":"9f2c4a7e3b1d8c9a6e5f4b2a1c3d7e8f"
}
```

The Wallet Instance MUST:

- Perform ICAO 9303 compliant NFC document reading (PACE,etc.).
- Validate document cryptographic signatures and certificate chains.
- Extract identity attributes (DG1 and DG11), Anti-Cloning Public Key from document data groups, and SODs (form MRTD and IAS Applications).
- Perform the Anti-Cloning Internal Authentication.
- Generate validation evidence in the JWT.
- Authenticate using a valid Wallet Instance Attestation.
- Include the exact `mrtd_auth_session` and `mrtd_pop_nonce` from the init response.
- Sign the `mrtd_validation_jwt` with its private key.
- Handle document reading errors and provide appropriate feedback.

The MRTD PoP Service MUST:

- Validate Wallet Instance Attestation according to IT-Wallet specifications.
- Verify OAuth-Client-Attestation-PoP signature.
- Validate the `mrtd_auth_session` parameter matches an active session.
- Verify the `mrtd_pop_nonce` matches the value sent in the previous response.

- Parse and validate the `mrtd_validation_jwt` signature (using Wallet Instance's public key) and structure.
- Extract and verify document evidence from the `mrtd_validation_jwt`.
- Validate document authenticity using ICAO 9303 standards.
- Verify Anti-Cloning challenge response.
- Validate document cryptographic proofs and certificate chains.
- Perform identity correlation between document data and LoA3 result.
- Check document validity (non revocation status).
- Verify the binding between IAS and MRTD applications by checking that the NUN extracted from DG1 is present (as hashed value) in the IAS SOD, and the DG1 itself is present (as hashed value) in the MRTD SOD. This dual verification ensures both applications reside on the same physical chip.

MRTD PoP Validation Response

Upon successful completion of all checks by the MRTD PoP Service, it MUST send to the Wallet Instance an HTTP Response with code *202 Accepted* including the following parameters:

Table 12.12: MRTD PoP Validation Response Parameters

Parameter	Type	Description
status	string	REQUIRED. It MUST be *require_interaction*.
type	string	REQUIRED. It MUST be *redirect_to_web*.
mrtd_val_pop_nonce	string	REQUIRED. New nonce for final confirmation.
redirect_uri	string	REQUIRED. Browser redirect URI for completion of the Authorization Flow.

Below a non-normative example of an MRTD PoP Validation Response:

```
HTTP/1.1 202 Accepted
Content-Type: application/json; charset=utf-8

{
  "status": "require_interaction",
        "type": "redirect_to_web",
  "mrtd_val_pop_nonce": "0f2bff024317345b6927ce17e776361d",
  "redirect_uri":"https://eaa-provider.example.org/cb"
}
```

The MRTD PoP Service MUST:

- Generate a new nonce `mrtd_val_pop_nonce` for browser-based final confirmation.
- Return HTTP 202 status to indicate async processing completion.

The Wallet Instance MUST:

- Validate the response.
- Extracts the parameters nonce `mrtd_val_pop_nonce` and `redirect_uri` to prepare the next browser-based GET Request.

Browser-based Final Confirmation

Upon successful MRTD PoP validation, the Wallet Instance MUST redirect the User Agent to the `challenge_redirect_uri` specified in the initial Authorization Details Object, including the `mrtd_val_pop_nonce` and `mrtd_auth_session` as a query parameter:

```
https://eaa-provider.example.org/l2plus-callback?mrtd_val_pop_nonce=0f2bff024317345b6927ce17e7
76361d_signed&mrtd_auth_session=wxroVrBY2MCq4dDNGXACS
```

The Wallet Instance MUST validate that the `mrtd_val_pop_nonce` matches the value received from the MRTD PoP Validation Response.

The Authorization Server MUST:

- Validate the `mrtd_val_pop_nonce` matches the value sent in the verification response, and that it is signed using the Wallet Instance's private key.
- Verify the `mrtd_auth_session` parameter matches an active session.
- Verify all authentication steps (LoA3 + MRTD PoP) have been completed successfully (including the identity correlation between LoA3 and document evidence).
- Generate the final authorization code.

Phase 4: OAuth Authorization Response

Upon successful completion of all authentication steps and identity matching, the Authorization Server MUST issue the final OAuth Authorization Response. If all validation checks have been passed, the Authorization Server MUST redirect the User Agent to the Wallet Instance with an OAuth Authorization Response including the authorization code as defined in steps 6-7 of *Low-Level Issuance Flow* (page 422), and in Section credential-issuance-endpoint:Authorization Response. The Authorization Server MUST use the `redirect_uri` included in the initial Request Object by the Wallet Instance.

Error Management

The error management MUST follow the same rules as defined in the Section *Credential Issuance Endpoints* (page 565), regarding the formats and the related standard references.

During the MRTD PoP Validation flow, when recoverable errors occur, the MRTD PoP Service MAY generate and return a fresh nonce to enable the User to retry attempts while maintaining session security and preventing replay attacks.

In addition to the error codes already defined in the Section *Credential Issuance Endpoints* (page 565), at least the following error codes MUST be supported.

MRTD PoP Response Errors

Table 12.13: MRTD PoP Response Error Codes

Error Code	Description
invalid_client	Wallet Instance authentication failed.
invalid_request	HTTP Request is invalid or malformed (malformed structure, missing data, etc.) or required session parameters are missing or invalid.
access_denied	User is not eligible for eID Substantial Authentication with MRTD Verification mechanism (e.g. CIE not found in the CIE Registry)
temporarily_unavailable	Document validation service or CIE Registry service is temporarily unavailable.

MRTD PoP Validation Response Errors

Table 12.14: MRTD PoP Validation Response Error Codes

Error Code	Description
invalid_client	Wallet Instance authentication failed.
invalid_request	HTTP Validation Request or Validation JWT is invalid or malformed (due to a malformed structure, missing data, signature failure, request timeout, etc.).
access_denied	User authentication or document validation failed.
invalid_document	Document cryptographic validation failed (SOD validation, IAS/MRTD binding, revocation status, etc.).
id_matching_failed	The matching between the identity obtained during primary authentication (eID LoA3) and the one obtained from the PoP of the Electronic Document failed.
temporarily_unavailable	Document validation service or CIE Registry service is temporarily unavailable.

HTTP Status Code Mapping

Error responses MUST use appropriate HTTP status codes:

- **400 Bad Request**: For `invalid_request` errors.
- **401 Unauthorized**: For `invalid_client` errors.
- **403 Forbidden**: For `access_denied` errors.
- **422 Unprocessable Entity**: For `invalid_document` or `id_matching_failed` errors.
- **503 Service Unavailable**: For `temporarily_unavailable` errors.

Security Considerations

Secure Session Management

The `mrtd_auth_session` parameter serves as the primary correlation mechanism between authentication steps. Implementations MUST ensure this identifier has sufficient entropy (minimum 128 bits) and is cryptographically secure. The session identifier MUST be validated at each step to prevent session fixation attacks.

Each authentication step MUST be cryptographically bound to the OAuth session through signed JWT validation to prevent session fixation, session confusion, and identity substitution attacks. The Authorization Server MUST maintain the correlation between the LoA3 identity and the document proof within a single session context.

In particular, each authentication step MUST validate the correlation between:

- Session identifier consistency across all protocol phases.
- LoA3 identity attributes binding to the session context.
- Document evidence correlation with authenticated identity.
- Temporal sequence validation to prevent out-of-order attacks.

When components operate outside the EAA Provider boundary, the following additional security measures MUST be implemented:

- Secure Inter-Service Communication (for example, through Certificate pinning, mutual TLS, etc.).
- Encryption and integrity of sensitive session data and/or personal identity information (for example, using JWE/JWS tokens).
- Distributed locks for session state updates.

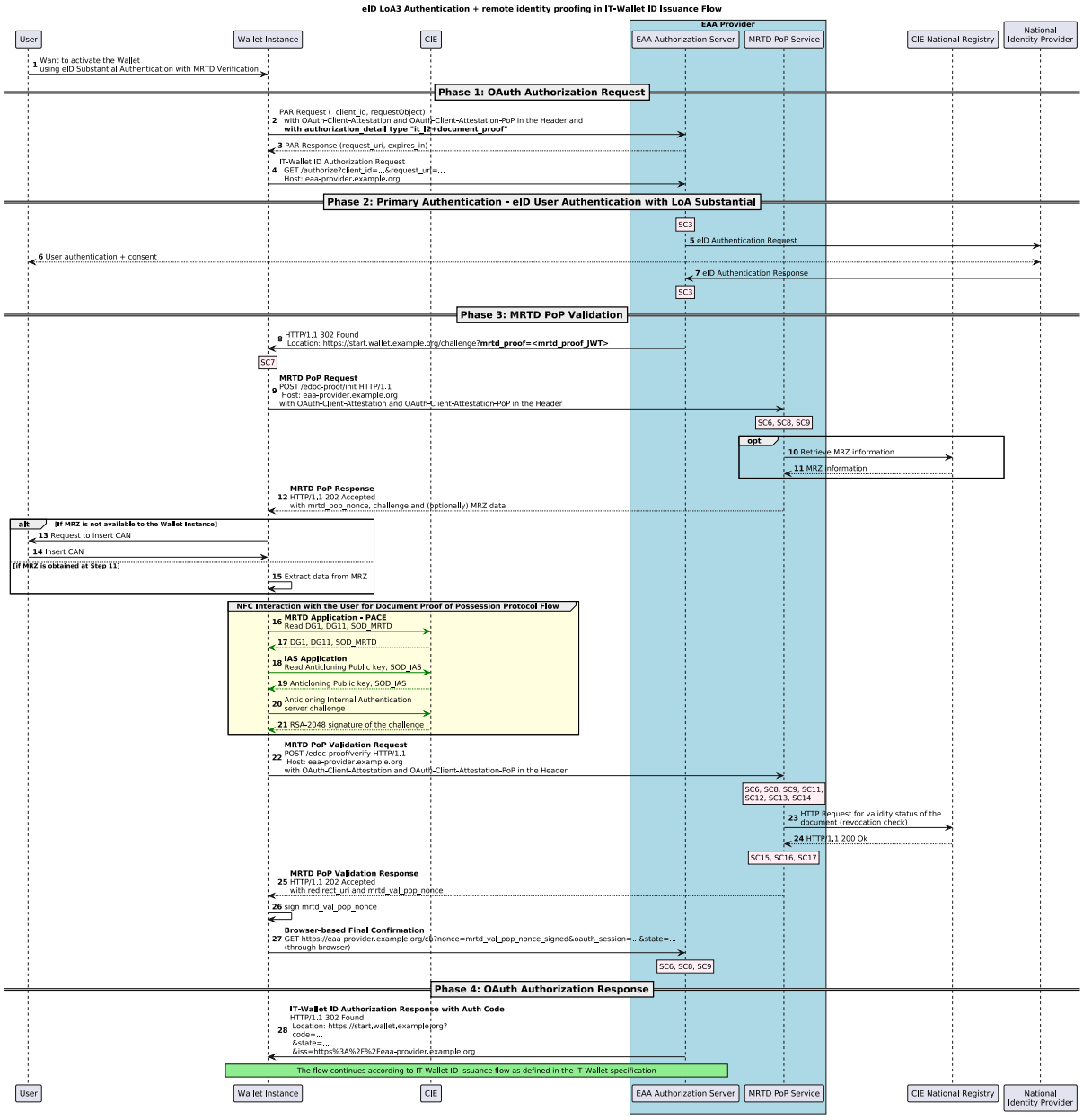


Fig. 12.11: eID Substantial Authentication with MRTD Verification Security Controls.

Cryptographic Challenge Generation

The MRTD PoP Service MUST generate cryptographically secure challenges with sufficient entropy (minimum 256 bits) for document validation. Challenge values MUST be unique and MUST NOT be reused across different sessions (cryptographically bound to the specific OAuth session context) or authentication attempts. The challenge generation algorithm SHOULD incorporate the `mrtd_auth_session` identifier and timestamp to ensure proper cryptographic binding.

Nonce Lifecycle Management

Each step in the eID Substantial Authentication with MRTD Verification flow MUST use unique nonce values (minimum 128 bits) to prevent replay attacks. Nonce values MUST have appropriate expiration times and MUST be invalidated after successful use. The EAA Authorization Server and MRTD PoP Service MUST maintain synchronized nonce validation to ensure session integrity.

Moreover, each nonce serves a specific security purpose. - `mrtd_pop_jwt_nonce` MUST be correlated with the MRTD Proof JWT. - `mrtd_pop_nonce` MUST:

- Be cryptographically independent from `mrtd_pop_jwt_nonce`.
- Incorporate the `mrtd_pop_jwt_nonce` as input to maintain the chain of trust.
- Use a different entropy source to prevent correlation attacks.
- `mrtd_val_pop_nonce` MUST:
- Be signed by the Wallet Instance private key.
- Include anti-replay timestamp validation.
- Be verified against the entire nonce chain for integrity.

EAA Provider Metadata

In addition to the `trust_frameworks_supported` values defined in section credential-issuer-metadata:Metadata for openid_credential_issuer, the EAA Provider Metadata for `openid_credential_issuer` MUST also support the value `it_l2+document_proof` indicating the multi-step Authentication protocol described in this Specification.

Security Controls

The following security controls MUST be implemented in the protocol:

Table 12.15: eID Substantial Authentication with MRTD Verification Security Controls

#	Description	Phase
SC1	The network communication channels are protected by using TLS with secure ciphers (at the time of this release, it is at least TLS v1.2).	All
SC2	The Wallet Instance ensures the authenticity of the EAA Provider and the eID Identity Provider by pinning the leaf certificate of each server.	All
SC3	The EAA Authorization Server verifies that the eID Identity Provider used within the *eID LoA3* phase is trustworthy.	Phase 2
SC4	The EAA Authorization Server verifies the authenticity and integrity of the data extracted from the *eID LoA3* assertion, by checking the digital signature.	Phase 2
SC5	The challenge value and all nonce values are generated with safeguards to prevent bruteforce or guessing attacks.	Phase 3
SC6	The EAA Provider verifies all the nonce values to detect replay attacks.	Phase 3
SC7	The Wallet Instance verifies that `challenge_info` is properly signed by the EAA Authorization Server. Moreover, it checks that `challenge_info` contains: an `iss` value corresponding to the value of the EAA Authorization Server; an `aud` value equal to the `client_id` of the Wallet Instance; and a `state` value equal to the one in the PAR request, to be sure that the response is bound to the initial request that is made by the Wallet Instance in Step 2. Therefore the information provided as part of `challenge_info`, in particular the `htu` that corresponds to the redirect url to follow for the *MRTD PoP*, is not tampered with.	Phase 3
SC8	The EAA Provider checks that the `mrtd_auth_session` is associated with the same Wallet Instance in all the requests within the *MRTD PoP* phase. Therefore the EAA Provider can be sure that the Wallet Instance performing the *MRTD PoP* phase: is trusted; is always the same across the protocol; and has previously started the IT-Wallet ID issuance (PAR request). This can be implemented by requesting the Wallet Instance to perform a proof of possession of its private key (e.g., within OAuth-Client-Attestation or by signing a nonce value).	Phase 3
SC9	The EAA Provider checks that the `mrtd_auth_session` is not expired (validity timeout typically of 5 minutes), i.e., that the operation has been concluded within a certain amount of time.	Phase 3
SC10	The integrity and confidentiality of the channel between the *physical CIE* and the *physical device_wallet* is secured with the PACE protocol (via the algorithm and key derivation functions supported by the card).	Phase 3
SC11	The MRTD PoP Service verifies the authenticity and integrity of the `mrtd_validation_jwt` by checking that it is signed with the Wallet Instance private key associated with the `mrtd_auth_session`.	Phase 3

Additional implementation requirements:

- **Rate limiting**: Protection against automated attacks and brute force attempts.
- **Session timeout**: Automatic cleanup of incomplete authentication sessions.
- **Audit logging**: Comprehensive logging of all authentication events with correlation identifiers.
- **Error handling**: Secure error responses that do not leak sensitive information.
- **Cryptographic material cleanup**: Secure deletion of temporary keys and challenges.

Implementation Considerations

Implementations MUST incorporate rate-limiting mechanisms to protect against automated attacks and resource exhaustion. Implementations SHOULD configure an interaction timeout that balances user experience and security posture, accommodating the variability inherent in NFC-based document reading.

The EAA Provider MUST implement session timeouts with cleanup mechanisms, ensuring session resources are released and temporary cryptographic material is securely deleted when sessions expire.

All security-relevant events throughout the eID Substantial Authentication with MRTD Verification flow MUST be logged with sufficient detail for auditing purposes while preserving the privacy of the User, ensuring that personally identifiable information, when stored, is appropriately hashed. The audit logs MUST have consistent correlation identifiers, enabling end-to-end tracing across all protocol phases, with cryptographic integrity protection to prevent tampering.

12.2 Digital Credential Presentation

This section describes how a Relying Party Instance requests to a Wallet Instance the presentation of the PID/EAAs.

In this section the following flows are described:

- *Remote Flow* (page 483), where the User presents a Digital Credential to a web Relying Party Instance according to OpenID4VP. In this scenario the user-agent and the Wallet Instance can be used in the same device (**Same Device Flow**), or in different devices (**Cross Device Flow**).
- *Proximity Flow* (page 511), where the User presents a Digital Credential to a mobile Relying Party Instance according to ISO18013-5. The User interacts with a Verifier using proximity connection technologies such as using QR Codes and Bluetooth Low Energy (BLE).

- *Identity Matching and Identity Reconciliation* (page 538), applied after cryptographic verification of a presented Digital Credential.

Proximity or otherwise offline presentation of a Digital Credential MUST succeed only when the bound private keys are stored in a local Keystore or a local WSCD. Digital Credentials whose bound private keys are stored in a remote WSCD MUST NOT be presented offline (*WP_160* (page 739)).

Note

When a Relying Party needs to distinguish the User during authentication, it MUST do so on the basis of the information contained in the Digital Credentials requested in the presentation request. The corresponding claims must be included in the presentation request. The Relying Party distinguishes the User by evaluating the values of the claims actually presented.

Non-normative examples are provided below.

1. **PID or IT-Wallet ID.** The Relying Party requests the identification claims defined by the relevant data model, for example `given_name`, `family_name` and `personal_administrative_number` for the PID, or `tax_id_code` for the IT-Wallet ID. See *PID Data Model* (page 380) and *IT-Wallet ID Data Model* (page 392).

   Example of a `dcql_query` in the remote flow (PID in SD-JWT VC format):

```
{
  "credentials": [
    {
      "id": "pid",
      "format": "dc+sd-jwt",
      "meta": {
        "vct_values": [ "urn:eudi:pid:it:1" ]
      },
      "claims": [
        {"path": ["given_name"]},
        {"path": ["family_name"]},
        {"path": ["personal_administrative_number"]}
      ]
    }
  ]
}
```

   The same claims, in the proximity flow, are requested in the mdoc `ItemsRequest`:

```
{
  "docType": "eu.europa.ec.eudi.pid.1",
  "nameSpaces": {
    "eu.europa.ec.eudi.pid.1": {
      "given_name": false,
```

```
        "family_name": false,
        "personal_administrative_number": false
      }
    }
  }
```

2. **Another Digital Credential from the catalog.** The Relying Party requests the claims defined by the schema of the Digital Credential published in the *Digital Credentials Catalog* (page 179). The corresponding data model is defined by the related schema in the *Schema Registry* (page 219).

   Example of a `dcql_query` in the remote flow for an mDL (`credential_type mDL`, mdoc-CBOR format, `docType org.iso.18013.5.1.mDL`):

```
{
  "credentials": [
    {
      "id": "mobile driving license",
      "format": "mso_mdoc",
      "meta": {
        "doctype_value": "org.iso.18013.5.1.mDL"
      },
      "claims": [
        {"path": ["org.iso.18013.5.1", "given_name"]},
        {"path": ["org.iso.18013.5.1", "family_name"]},
        {"path": ["org.iso.18013.5.1", "document_number"]}
      ]
    }
  ]
}
```

Note

In the case of using the batch Credential, the Wallet Instance SHOULD implement a Credential selection logic (e.g., based on the earliest expiring one) and MUST mark it as consumed. At the end of the flow it MUST decrease the number of available Credentials in the batch and using that it can control the time when it needs to obtain a new batch of Credentials.

12.2.1 Remote Flow

Depending on whether the User is using a mobile device or a workstation, the Relying Party MUST support the following remote flows (RPR-84):

- **Same Device**, the Relying Party MUST provide an `HTTP` location to the Wallet Instance using a redirect (`302`) or an HTML href in a web page (RPR-01);
- **Cross Device**, the Relying Party MUST provide a `QR Code` which the User frames with the device camera or with the Wallet Instance (RPR-03).

To invoke the correct Wallet Instance, the Relying Party SHOULD trigger the Wallet Instance installed on the User's device the User wishes to use. This information SHOULD be provided by the User using the Selection Page described in *User Experience Design* (page 35).

- If the Selection Page is supported, the User selects the Wallet, and then the Relying Party retrieves the Wallet metadata as described in *Wallet Metadata Retrieval Flow* (page 541). The content of the HTML href or QR Code depends on the `authorization_endpoint` parameter in the Wallet metadata:
  - If `authorization_endpoint` is available and contains an HTTPS URL (Universal Link), the Relying Party SHOULD use that endpoint.
  - Otherwise, the Relying Party MUST use one of the custom URL schemes: `openid4vp://` (as defined in Section 13.1.2 of [OpenID4VP]) or `haip-vp://` (as defined in Section 5.1 of [OPENID4VC-HAIP]). The Wallet Instance MUST support both custom URL schemes.
- In the case when the Relying Party does not support the Selection Page, or the retrieval of the Wallet metadata may fail for some reason, the Relying Party invokes the Wallet Instance using the custom URL scheme described above.

After the Wallet Instance invocation, the Wallet Instance establishes the trust with the Relying Party and evaluates the request. If valid, it will ask Users to give their consent for the disclosure of the Digital Credentials, in the form of a Verifiable Presentation.

A High-Level description of the remote flow, from the User's perspective, is given below and shown in *High Level Remote Protocol Flow.* (page 484):

1. *Authorization Request*: the Wallet Instance obtains a `URL` in the Same Device flow or a `QR Code` in Cross Device flow containing the signed Request Object either
   - directly, by passing a Request Object by value (via `request` parameter) or
   - by passing a Request Object by a reference (via `request_uri` parameter), where the signed Request Object is available for download. This

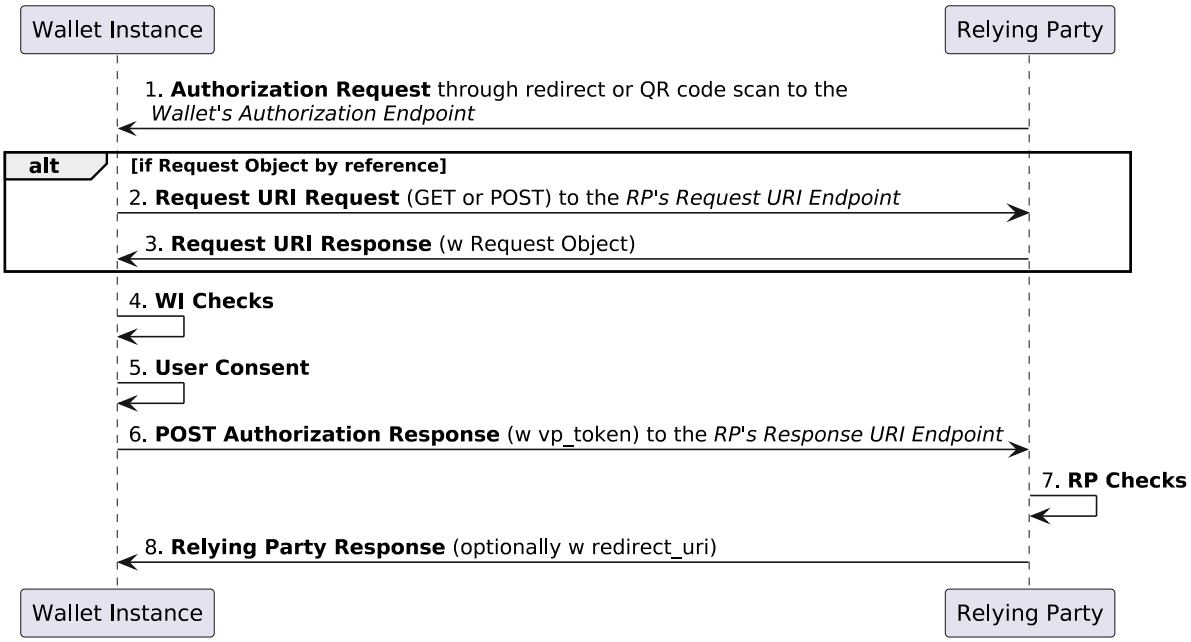


Fig. 12.12: High Level Remote Protocol Flow.

is the only method permitted for OpenID4VC High Assurance Interoperability Profile (HAIP)-Compliant Relying Parties OPENID4VC-HAIP.

If by reference, Steps 2 and 3 are performed.

2. *Request URI Request*: the Wallet Instance extracts from the payload the following parameters: `client_id`, `request_uri`, `request_uri_method`.

- If `request_uri_method` is provided and set with the value `post`, the Wallet Instance SHOULD transmit its metadata to the Relying Party's `request_uri` endpoint using the `HTTP POST` method.
- If `request_uri_method` is set with the value `get` or not present, the Wallet Instance MUST fetch the signed Request Object using an `HTTP` request with method `GET` to the endpoint provided in the `request_uri` parameter (RPR-08).

3. *Request URI Response*: the Relying Party returns a signed Request Object to the Wallet Instance.

4. *WI Checks*: the Wallet Instance:

a. verifies the signature of the signed Request Object using the public key identified in the `JWT` header of the Request Object. Using that reference, the Wallet Instance is able to select the correct Relying Party's public key for signature verification (*WP_085* (page 754)).

b. verifies that the `client_id` contained in the Request Object issuer (Relying Party) matches with the one obtained at the Step 2:

   - If `client_id` uses the `openid_federation` prefix, it MUST match the `sub` parameter contained in the Relying Party's Entity Configuration within the Trust Chain (*WP_086* (page 754)).
   - If `client_id` uses the `x509_hash` prefix, the Wallet Instance MUST verify that the hash of the Relying Party's X.509 certificate (in the `x5c` request header) matches the hash contained in `client_id` from Step 2 (as defined in OpenID4VP, Section 5.9.3).

c. evaluates the requested Digital Credentials and checks the eligibility of the Relying Party in asking for these by applying the policies related to that specific Relying Party, obtained with the Trust Chain (*WP_087* (page 754)).

5. *POST Authorization Response*: the Wallet Instance presents the requested information to the Relying Party.
6. *RP Checks*: The Relying Party validates the presented Credentials by verifying the trust with their Issuers and checks the respective validity status.
7. *Relying Party Response*: the Wallet Instance informs the User about the successful authentication with the Relying Party, and the User continues the navigation.

Below is a sequence diagram that details the interactions between all the involved parties.

The details of each step shown in the previous picture are described below.

Steps 1-2: The User requests to access to a protected resource of the Relying Party.

Step 3: The Relying Party creates a fresh, cryptographically random state value with sufficient entropy, binds it to the user-agent session (e.g., using an HTTP secured cookie), and stores it server-side with a short expiration time. It then inspects the user-agent to determine whether the flow occurs on the same device as the user-agent.

Steps 4-7 (Authorization Request): The Relying Party provides the user-agent with a JavaScript page inspecting the status endpoint and the Wallet Instance with a URL containing the Authorization Request.

> In the **Cross Device Flow**, the Authorization Request is presented as a QR Code displayed to the User. The User scans the QR Code using the Wallet Instance and retrieves a URL. Below is represented a non-normative example of a QR Code issued by the Relying Party.

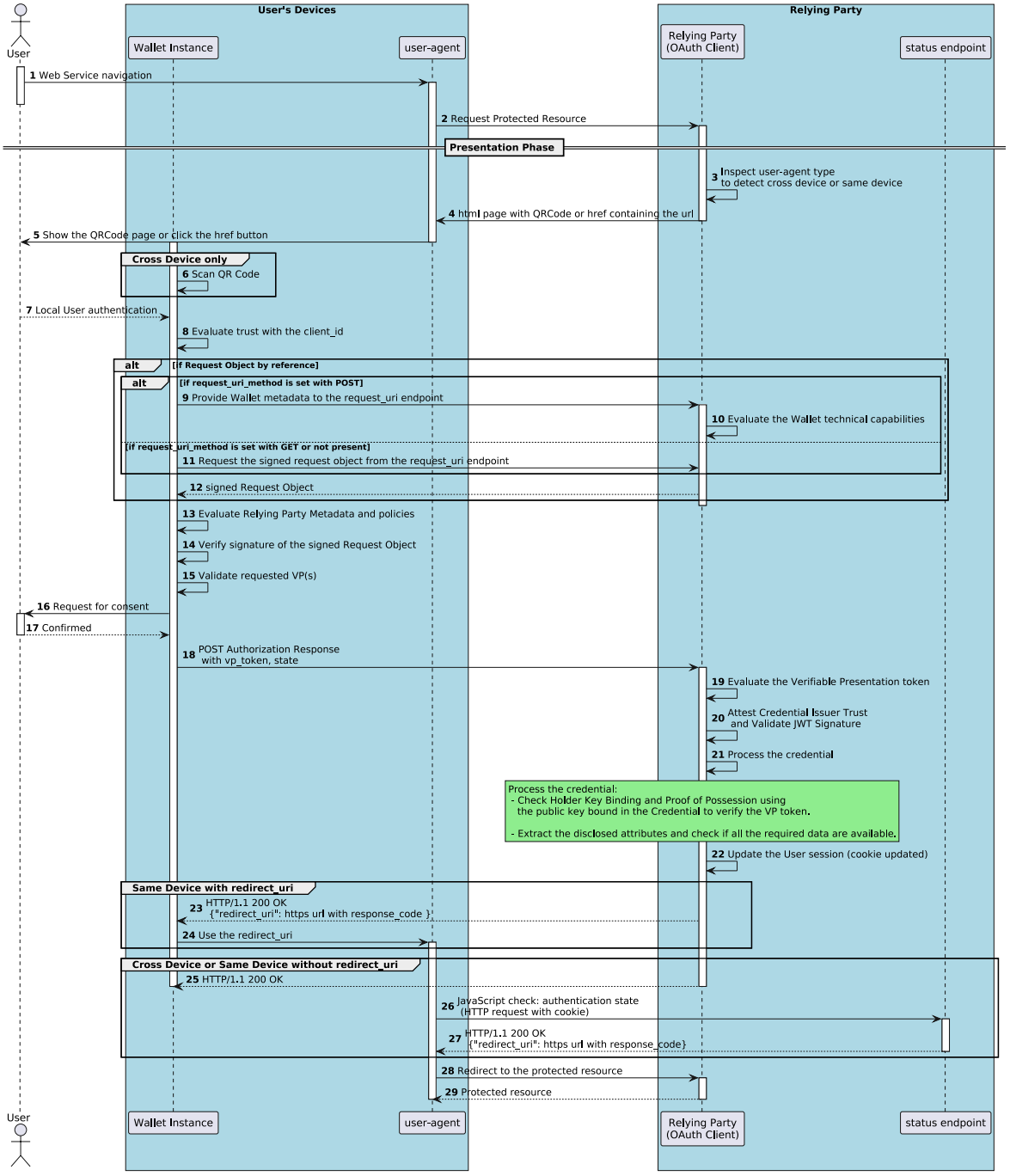


Fig. 12.13: Remote Protocol Flow.



Note

The *error correction level* chosen for the QR Code MUST be Q (Quartily - up to 25%), since it offers a good balance between error correction capability and data density/space. This level of quality and error correction allows the QR Code to remain readable even if it is damaged or partially obscured (RPR-77).

If the Request Object is passed by value the URL inside the QR Code contains `client_id` and `request` parameters. Below a non-normative example of the QR Code raw payload with Request Object by value:

```
https://wallet-solution.example.org/authorization?client_id=openid_federation%3Ahttps%3A%
2F%2Frelying-party.example.org&request=eyJhbGciOiJFUzI1NiIs...9t2LQ
```

While if the Request Object is passed by reference the URL inside the QR Code contains `client_id`, `request_uri`, and `request_uri_method` parameters (*WP_076–077* (page 754)). Below is represented a non-normative example of the QR Code raw payload with Request Object by reference:

```
https://wallet-solution.example.org/authorization?client_id=openid_federation%3Ahttps%3A%
2F%2Frelying-party.example.org&request_uri=https%3A%2F%2Frelying-party.example.org&reques
t_uri_method=post
```

An official, self-contained HTML template for this **Cross Device** QR code page—including header, footer, accessibility, multilingual copy, and a configurable demonstrative payload—is provided in the official-resources:HTML Components section (**IT-Wallet Presentation QR Code Page**). It is linked from the wallet cards on the **IT-Wallet Selection Page** in the same section.

Conversely, in the **Same Device Flow**, the Relying Party uses an HTTP response redirect (with status code set to 302) or an html page with an href button, containing the URL providing the same information as in the Cross-Device Flow (*WP_076–077* (page 754)). Below is a non-normative example with Request Object by reference:

```
HTTP/1.1 302 Found
Location: https://wallet-solution.digital-strategy.europa.eu?client_id=openid_federation%3
Ahttps%3A%2F%2Frelying-party.example.org%2Fcb&request_uri=https%3A%2F%2Frelying-party.exam
ple.org%2Frequest_uri&request_uri_method=post
```

Step 8: The Wallet Instance evaluates the trust with the Relying Party (*WP_078–080* (page 754)).

Steps 9-11 (Request URI Request): The Wallet Instance checks if the Relying Party has provided the `request_uri_method` within its signed Request Object (*WP_083* (page 754)).

- If it is provided and is equal to `post`, the Wallet Instance SHOULD provide its metadata to the Relying Party. The Relying Party updates the Request Object according with the Wallet technical capabilities.

  The following is a non-normative example of an HTTP request made by the Wallet Instance to the Relying Party.

```
POST /request HTTP/1.1
Host: client.example.org
Content-Type: application/x-www-form-urlencoded
Accept: application/oauth-authz-req+jwt

wallet_metadata=%7B%22authorization_endpoint%22%3A%20%22https%3A%2F%2Fwallet-solution.digi
tal-strategy.europa.eu%2Fauthorization%22%2C%20%22response_types_supported%22%3A%20%5B%22v
p_token%22%5D%2C%20%22vp_formats_supported%22%3A%20%7B%22dc%2Bsd-jwt%22%3A%20%7B%22sd-jwt_
alg_values%22%3A%20%5B%22ES256%22%2C%20%22ES384%22%5D%7D%2C%22mso_mdoc%22%3A%7B%22issuerau
th_alg_values%22%3A%5B-9%2C-51%5D%2C%22deviceauth_alg_values%22%3A%5B-9%2C-51%5D%7D%7D%2C%
22request_object_signing_alg_values_supported%22%3A%20%5B%22ES256%22%5D%2C%22client_id_pre
fixes_supported%22%3A%5B%22openid_federation%22%2C%22x509_hash%22%5D%7D%26wallet_nonce%3Dq
PmxiNFCR3QTm19POc8u
```

  Where the body of the request prior to being encoded in *application/x-www-form-urlencoded* by the Wallet corresponds to:

```
{
  "wallet_metadata": {
    "authorization_endpoint": "https://wallet-solution.digital-strategy.europa.eu/authoriz
ation",
    "response_types_supported": [
      "vp_token"
    ],
    "vp_formats_supported": {
      "dc+sd-jwt": {
```

```
            "sd-jwt_alg_values": ["ES256", "ES384"]
        },
        "mso_mdoc": {
            "issuerauth_alg_values": [-9, -51],
            "deviceauth_alg_values": [-9, -51]
        }
      },
      "request_object_signing_alg_values_supported": ["ES256"],
      "client_id_prefixes_supported": ["openid_federation", "x509_hash"]
    },
    "wallet_nonce": "qPmxiNFCR3QTm19POc8u"
}
```


- When the Wallet Instance capabilities discovery is not supported by Relying Party, the Wallet Instance requests the signed Request Object using the HTTP method GET (*WP_082* (page 754)).

Step 12 (Request URI Response): The Relying Party issues the Request Object signing it using one of its cryptographic private keys, where their public parts have been published within its Entity Configuration (*metadata.openid_credential_verifier.jwks*) as extracted by Wallet Instance per *WP_084* (page 754). The Wallet Instance obtains the signed Request Object.

Below is a non-normative example of the Redirect URI Response:

```
HTTP/1.1 200 OK
Content-Type: application/oauth-authz-req+jwt

eyJhbGciOiJFUzI1NiIs...9t2LQ
```

Steps 13-15 (WI Checks): The Wallet Instance verifies the Request Object, which is in the form of a signed JWT (*WP_085–086* (page 754)).

A non-normative example of a Request Object in the form of decoded header and payload is shown below:


```
{
  "alg": "ES256",
  "typ": "oauth-authz-req+jwt",
  "kid": "9tjiCaivhWLVUJ3AxwGGz_9",
  "trust_chain": [
    "MIICajCCAdOgAwIBAgIC...awz",
    "MIICajCCAdOgAwIBAgIC...2w3",
    "MIICajCCAdOgAwIBAgIC...sf2"
  ],
  "x5c": [
    "MIIDqjCCApKgAwIBAgIESLNEvDA ...",
    "MIICwzCCAasCCQCKVy9eKjvi+jA ...",
    "MIIDTDCCAjSgAwIBAgIJAPlnQYH..."
```

```
    ]
}

{
  "client_id": "openid_federation:https://relying-party.example.org",
  "response_mode": "direct_post.jwt",
  "response_type": "vp_token",
  "dcql_query": {
    "credentials": [
      {
        "id": "personal id data",
        "format": "dc+sd-jwt",
        "meta": {
          "vct_values": [ "urn:eudi:pid:it:1" ]
        },
        "claims": [
          {"path": ["given_name"]},
          {"path": ["family_name"]},
          {"path": ["birthdate"]}
        ]
      },
      {
        "id": "mobile driving license",
        "format": "mso_mdoc",
        "meta": {
          "doctype_value": "org.iso.18013.5.1.mDL"
        },
        "claims": [
          {"path": ["org.iso.18013.5.1", "given_name"]},
          {"path": ["org.iso.18013.5.1", "family_name"]},
          {"path": ["org.iso.18013.5.1", "document_number"]}
        ]
      }
    ]
  },
  "response_uri": "https://relying-party.example.org/response_uri",
  "nonce": "2c128e4d-fc91-4cd3-86b8-18bdea0988cb",
  "wallet_nonce": "qPmxiNFCR3QTm19POc8u",
  "client_metadata": {
      "jwks": {
        "keys": [
          {
            "kty": "EC",
            "use": "enc",
            "crv": "P-256",
            "x": "f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
            "y": "x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
```

```
            "kid": "20260202-abc123",
            "alg": "ECDH-ES"
          }
        ]
      },
      "encrypted_response_alg_values_supported": ["ECDH-ES"],
      "encrypted_response_enc_values_supported": ["A128GCM", "A256GCM"]
  },
  "state": "3be39b69-6ac1-41aa-921b-3e6c07ddcb03",
  "iss": "https://relying-party.example.org",
  "iat": 1672418465,
  "exp": 1672422065
}
```


It then processes the Relying Party metadata and applies the relevant policies to determine which Digital Credentials and User data the Relying Party is authorized to request (*WP_087* (page 754)).

Steps 16-17 (User Consent): The Wallet Instance requests the User's consent to disclose the requested Credentials by showing the Relying Party's identity and the requested attributes. The User authorizes and consents the presentation of the Credentials by selecting and or deselecting the personal data to release (*WP_088* (page 754)).

Step 18 (Authorization Response): The Wallet Instance provides the Authorization Response to the Relying Party using an HTTP request with the method POST using response mode "direct_post.jwt".

Below is a non-normative example of the Authorization Response:


```
POST /response_uri HTTP/1.1
HOST: relying-party.example.org
Content-Type: application/x-www-form-urlencoded

response=eyJhbGciOiJFQ0RILUVTIiwiZW5jIjoiQTI1NkdDTSIsImtpZCI6ImVwaGVtZXJhbC0yMDI2MDIwMi1hY
mMxMjMiLCJlcGsiOnsi...fX0..5vL9d2X8fQ..dGhpcy1pcy1hLXNhbXBsZS1jaXBoZXJ0ZXh0.ABCDEFGHIJKLMN
OPQRS
```


Below is a non-normative example showing the decrypted JWE protected header and the payload of the JWT contained in the response, before base64url encoding. The `vp_token` parameter value corresponds to the format used when the DCQL query language is used in the presentation request.


```
{
  "alg": "ECDH-ES",
  "enc": "A256GCM",
  "kid": "20260202-abc123",
  "epk": {
    "kty": "EC",
```

```
      "crv": "P-256",
      "x": "f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
      "y": "x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0"
    }
  }

  {
    "state": "3be39b69-6ac1-41aa-921b-3e6c07ddcb03",
    "vp_token": {
      "personal id data": ["eyJhbGciOiJFUzI1NiIs...PT0iXX0"],
      "mobile driving license": ["o2Nkb2N0eXBlb3Jzby4xO...Nib3JfZHVtbXk"]
    }
  }
```


Note

When returning a requested Credential in `mso_mdoc` format in the `vp_token`, the Wallet MUST cryptographically bind the resulting mdoc presentation to the current OpenID4VP transaction. To achieve this, the Wallet builds the ISO `SessionTranscript` used for mdoc device authentication and applies the OpenID4VP profiling rules by setting `DeviceEngagementBytes` to `null` and `EReaderKeyBytes` to `null`, and sets its `Handover` field to an OpenID4VP-defined structure (`OpenID4VPHandover`) derived from the Authorization Request parameters. The Wallet then computes the mdoc device authentication (device signature) over data that includes this `SessionTranscript`, such that the resulting mdoc presentation is valid only for that specific OpenID4VP transaction. For the normative definition of `OpenID4VPHandover` and the corresponding `SessionTranscript` profiling rules, see OpenID4VP Appendix B.2.

Steps 19-22 (RP Checks): The Relying Party verifies the Authorization Response, extracts the `vp_token`, which contains one or more Digital Credentials presentations, and validates the overall format of the VP Token. For each credential presentation, the Relying Party verifies its integrity according to the DCQL query criteria defined in the Authorization Request. The Relying Party MUST also attest trust with the corresponding Credentials Issuer, and validate the Wallet Instance's proof of possession of each presented Digital Credential. Finally, the Relying Party verifies the revocation status of each presented Digital Credential, as described in *Digital Credential Revocation and Suspension* (page 404). If all previous verifications yielded positive result, the Relying Party updates the User session.

Steps 23-24 or 25 (Relying Party Response): The Relying Party provides to the Wallet Instance the response about the presentation, which informs the User.

> Upon receiving and validating the Authorization Response at the Response Endpoint, the Relying Party returns to the Wallet Instance an HTTP 200 OK. In particular, in the Same Device Flow, the Relying Party MUST also pass the `redirect_uri` parameter in the response to the Wallet Instance. Upon receiving the `redirect_uri`,

the Wallet Instance MUST perform a redirect to the URL specified by the `redirect_uri`. This redirect allows the Relying Party to seamlessly resume interaction with the User on the device which initiated the flow. When the response does not contain the `redirect_uri` parameter, the Wallet Instance is not required to perform any further step. The User should manually close the Wallet Instance and open the user-agent to continue the flow (RPR-83).

The following is a non-normative example of the response in the Same Device Flow.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "redirect_uri": "https://relying-party.example.org/cb?response_code=091535f699ea575c7937fa5f0
f454aee"
}
```

Steps 26-27: The JavaScript page is inspecting the status endpoint.

Below is a non-normative example of the HTTP Request to the status endpoint, where the parameter `id` contains an opaque and random value:

```
GET /session-state?id=3be39b69-6ac1-41aa-921b-3e6c07ddcb03 HTTP/1.1
HOST: relying-party.example.org
```

When the Wallet Instance has provided the presentation to the Relying Party's **response_uri** endpoint and, in the Same Device Flow, the user-agent has successfully returned via `redirect_uri` within the same user session, the User authentication is successful. The Relying Party updates the session cookie allowing the user-agent to access to the protected resource. A redirect URL is provided carrying the location where the user-agent is intended to navigate. The following is a non-normative example of the response with the `redirect_uri` from the Relying Party to the user-agent.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "redirect_uri": "https://relying-party.example.org/cb?response_code=091535f699ea575c7937
fa5f0f454aee"
}
```

Steps 28-29: The user-agent is redirected to the redirect URI to continue the navigation with the protected resource made available to the User (*WP_094* (page 754)). The Relying Party MUST consider the transaction completed only if the redirect back is received in the same user session in which the flow was initiated; otherwise it MUST reject the presentation.

Note

During each credential presentation transaction executed through the remote flow, the Wallet Instance MUST create and maintain a corresponding transaction record in the transaction log (see *Wallet Instance Dashboard and Transaction Logging* (page 299)).

The transaction record MUST be created once the Wallet Instance has accepted the presentation request for processing (i.e., after request validation and Relying Party trust/policy checks, Steps 13–15). At this point, the record MUST include the transaction metadata and the request context available at that stage (e.g., the requested Credential type(s) and the identifier(s) of the requested attributes), without logging any attribute values.

The record MUST be updated as the transaction progresses to reflect the evolving transaction state and result context (e.g., what was actually presented after User consent and response preparation/sending, Steps 16–18), without logging any attribute values.

The record MUST be finalized when the transaction ends, indicating the outcome (e.g., completed, failed, or aborted; Steps 23–29).

Authorization Request

The URL parameters contained in the Relying Party Authorization Request are described in the table below.

Name	Description
client_id	REQUIRED. Unique identifier of the Relying Party. The value MUST use one of the following Client Identifier Prefixes (as defined in OpenID4VP, Section 5.9): `openid_federation` (Relying Party's Entity Identifier in a Trust Chain) or `x509_hash` (base64url-encoded SHA-256 hash of the Relying Party's X.509 certificate).
request	CONDITIONAL. REQUIRED unless `request_uri` is specified. It contains the base64url-encoded and signed Request Object. For the content of the Request Object see Section *Request Object* (page 498).
request_uri	CONDITIONAL. REQUIRED unless `request` is specified. The HTTP URL where the Relying Party provides the signed Request Object to the Wallet Instance.

continues on next page

Table 12.16 – continued from previous page

Name	Description
request_uri_method	OPTIONAL only if `request_uri` is specified, otherwise MUST NOT be present. The HTTP method MUST be set with `get` or `post` (RPR-07, RPR-08, RPR-09). The Wallet Instance should use this method to obtain the signed Request Object from the `request_uri`. If not provided or equal to `get`, the Wallet Instance SHOULD use the HTTP method `get`. Otherwise, the Wallet Instance SHOULD provide its metadata within the HTTP POST body encoded in `application/x-www-form-urlencoded`.

Note

IT Wallet specification recommends the use of `request_uri`, i.e. Request Object by reference.

Warning

For security reasons and to prevent endpoint mix-up attacks, the value contained in the `request_uri` parameter MUST be one of those attested by a trusted third party, such as those provided in the `openid_credential_verifier` metadata within the `request_uris` parameter, obtained from the Trust Chain about the Relying Party (*WP_081* (page 754) and RPR-85).

The value corresponding to the `request_uri` endpoint SHOULD be randomized, according to RFC 9101, The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR) Section 5.2.1.

Request URI Request

The Relying Party SHOULD provide the POST method with its `request_uri` endpoint allowing the Wallet Instance to inform the Relying Party about its technical capabilities.

This feature can be useful when, for example, the Wallet Instance supports a restricted set of features, supported algorithms or a specific url for its `authorization_endpoint`, and any other information that it deems necessary to provide to the Relying Party for interoperability.

Warning

The Wallet Instance, when providing its technical capabilities to the Relying Party, MUST NOT include any User information or other explicit (RPR-86) information regarding the hardware used or usage preferences of its User (RPR-86).

If both the Relying Party and the Wallet Instance support the `request_uri_method` with HTTP POST, the Wallet Instance capabilities (metadata) MUST be provided using an HTTP request to the `request_uri` endpoint of the Relying Party, with the method POST and content type set to `application/x-www-form-urlencoded` (*WP_083* (page 754)). The request and its parameters are defined in Section 5 (Authorization Request) of OpenID4VP. Below are the normative details and references about the parameters to be used by the Wallet Instance in the request (*WP_083a–083c* (page 754)).

Table 12.17: Request URI Endpoint Parameters

Parameter	Description
wallet_metadata	OPTIONAL. JSON object with metadata parameters. See OpenID4VP, Section 10.1 and the table below, "Wallet Metadata Parameters".
wallet_nonce	RECOMMENDED. String used by Wallet Instance to prevent replay of the Relying Party's responses.

Table 12.18: Wallet Metadata Parameters

Parameter	Description
vp_formats_supported	REQUIRED. Object containing a list of name/value pairs, where the name is a Credential Format Identifier and the value defines format-specific parameters that a Wallet supports. See OpenID4VP Appendix B. Wallet Instances MUST support the Credential Format Identifiers required by OPENID4VC-HAIP (including `dc+sd-jwt` and `mso_mdoc`).
client_id_prefixes_supported	RECOMMENDED. A non-empty array of the Client Identifier Prefixes that the Wallet Instance supports. Valid values include `openid_federation` and `x509_hash`; if omitted, the default is `pre-registered`.
authorization_endpoint	URL of the authorization server's endpoint, see OAUTH2. Using a universal link is preferable for enhanced security and fallback support, custom url schemes can also be used if necessary.
response_types_supported	OPTIONAL. JSON array of OAuth 2.0 `response_type` values. If present it MUST be set to `vp_token` (RPR-82).
request_object_signing_alg_values_supported	OPTIONAL. See OpenID Connect Discovery.

Note

In the IT Wallet, legacy Relying Parties using an `https` URI as `client_id` implicitly follow the OpenID Federation client identifier prefix (`openid_federation`). Their trust is established and validated through trust chain resolution, which is treated as equivalent to that of statically trusted (pre-registered) clients as defined in [**RFC 6749**], for backward compatibility.

Note

The `wallet_nonce` parameter is RECOMMENDED for Wallet Instances that want to prevent reply of their http requests to the Relying Parties. When present, the Relying Party MUST evaluate it (RPR-81).

Note

For the `authorization_endpoint` the use of universal links are preferred over custom url-schemes because, when properly configured using Assetlinks JSON for Android and Apple App Site Association for iOS, they provide enhanced security by reducing the risk of URL hijacking. Furthermore, universal links offer fallback mechanisms, allowing the flow to continue seamlessly in a browser even if the Wallet Instance is not installed, ensuring a smoother User experience. The URL schemes `openid4vp://` and `haip-vp://` defined in OPENID4VP and OPENID4VC-HAIP are supported to ensure interoperability.

Request URI Response

The Relying Party issues the signed Request Object using the content type set to `application/oauth-authz-req+jwt`. For the content of the Request Object see Section *Request Object* (page 498).

Request URI Endpoint Errors

When the Relying Party encounters errors while issuing the Request Object from the `request_uri` endpoint, it MUST return an error response with `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`400 Bad Request`	`invalid_request`	The Request Object could not be retrieved due to an invalid or malformed request at the `request_uri` endpoint. (**RFC 6749#section-4.1.2.1**).

continues on next page

Table 12.19 – continued from previous page

Status Code	Error Code	Description
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Request URI Endpoint encountered an internal problem. (**RFC 6749#section-4.1.2.1**).
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Request URI Endpoint is temporarily unavailable (e.g., due to maintenance or overload). (**RFC 6749#section-4.1.2.1**).

The following is an example of an error response from `request_uri` endpoint:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
  "error": "server_error",
  "error_description": "The Request Object cannot be retrieved due to an internal server error.
"
}
```

Upon receiving an error response, the Wallet Instance SHOULD inform the User of the error condition in an appropriate manner (*WP_089* (page 754)). The Wallet Instance SHOULD log the error and MAY attempt to recover from certain errors if feasible (*WP_089a* (page 754)). For example, if the error is `server_error`, the Wallet Instance SHOULD prompt the User to re-enter or scan a new QR code, if applicable (*WP_089b* (page 754)).

Request Object

The JWT header parameters are described below:

Name	Description
alg	REQUIRED. Algorithm used to sign the JWT, according to [**RFC 7516#section-4.1.1**]. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or to a symmetric algorithm (MAC) identifier (RPR-88).
typ	REQUIRED. Media Type of the JWT, as defined in [**RFC 7519**] and [**RFC 9101**]. It SHOULD be set to the value `oauth-authz-req+jwt` (RPR-89).
kid	REQUIRED. Key ID of the public key needed to verify the JWT signature, as defined in [**RFC 7517**].
trust_chain	OPTIONAL. It is a sequence of Entity Statements that composes the Trust Chain related to the Relying Party, as defined in OIDFED Section 4.3 *Trust Chain Header Parameter*.

continues on next page

Table 12.20 – continued from previous page

Name	Description
x5c	REQUIRED when `client_id` uses an `x509_hash` prefix scheme. OPTIONAL when `client_id` uses the `openid_federation` scheme. It contains the X.509 certificate chain about the Relying Party, excluding the Trust Anchor certificate. This certificate MUST be used to verify the JWT signature. The Relying Party's certificate in `x5c` asserts the Relying Party identity information along with the network endpoints used in the presentation flow, including the endpoints Authorization Request and Response endpoints (`response_uri` and `redirect_uri`). All the endpoints used in the presentation flow MUST be bound to the FQDN and any further webpath provided in the Relying Party's certificate, in the form of URI-type SAN for full-URI matching, or a DNSName SAN for host-name matching.

Note

The `x5c` header MUST NOT include the root certificate, as required by OPENID4VC-HAIP. The `x5c` certificate chain MUST validate to a preconfigured root certificate; see Section *X.509 PKI* (page 138) for background on X.509 certificate chain validation.

The JWT payload parameters are described herein:

Name	Description
client_id	REQUIRED. Unique Identifier of the Relying Party.

continues on next page

Table 12.21 – continued from previous page

Name	Description
client_metadata	**REQUIRED. A JSON object containing the Relying Party metadata values as defined in Section 5.1 of OpenID4VP, that SHOULD include the following parameters:** • **vp_formats_supported**. Used by the Wallet Instance to determine the supported Verifiable Presentation formats. • **encrypted_response_enc_values_supported**. JSON array listing the supported JWE `enc` algorithms for encrypted Authorization Responses in `direct_post.jwt`. • **jwks**. JSON Web Key Set used by the Wallet Instance for encrypting the Authorization Response or for key agreement. Keys contained in this set are request-specific and identified by their `kid` value. • **client_name** and **logo_uri**. OPTIONAL. Used for user consent display and to show the Relying Party identity in the Wallet Instance interface.
response_mode	REQUIRED. It MUST be set to `direct_post.jwt` in both Same Device and Cross Device flows (RPR-90).
dcql_query	REQUIRED. Object representing a request for a presentation of Credentials, according to the DCQL query language defined in Section 6 of OpenID4VP.
transaction_data	**OPTIONAL. Non-empty array of JSON objects, each describing a transaction that the Relying Party requests the User to authorize. Each transaction object includes:** • **type**. String that identifies the transaction data type. • **credential_ids**. Array referencing one or more Credentials from the `dcql_query` that can authorize the transaction.
transaction_data_hashes_alg	OPTIONAL. Array of strings, each representing a hash algorithm identifier, corresponding to a hash algorithm name listed in the IANA. One of these algorithms MUST be used to calculate the hashes in the `transaction_data_hashes` response parameter. If omitted, the default hash algorithm is `sha-256`.
response_type	REQUIRED. It MUST be set to `vp_token` (RPR-91).
wallet_nonce	REQUIRED if previously provided by Wallet Instance (RPR-81). String value used to mitigate replay attacks of the response, as defined in Section 5.10 (Request URI Method) of OpenID4VP.

continues on next page

Table 12.21 – continued from previous page

Name	Description
response_uri	REQUIRED. The Response URI to which the Wallet Instance MUST send the Authorization Response using an HTTP request using the method POST (RPR-92).
nonce	REQUIRED. Fresh cryptographically random number with sufficient entropy, which length MUST be at least 32 digits (RPR-93).
state	RECOMMENDED. Unique identifier of the Authorization Request, its value SHOULD be opaque to the Wallet Instance.
iss	REQUIRED. The entity that has issued the JWT. It will be populated with the Relying Party client id.
iat	REQUIRED. Unix Timestamp, representing the time at which the JWT was issued.
exp	REQUIRED. Unix Timestamp, representing the expiration time on or after which the JWT MUST NOT be valid anymore (RPR-94).

Warning

For security reasons and to prevent endpoint mix-up attacks, the value contained in the `response_uri` parameter MUST be one of those attested by a trusted third party, such as those provided in the `openid_credential_verifier` metadata within the `response_uris` parameter, obtained from the Trust Chain about the Relying Party (*WP_091a* (page 754) and RPR-95).

Note

The `transaction_data` parameter is intended for use cases where the Wallet Instance MUST authorize a specific transaction, such as payment initiation or digital signing. In these high-sensitivity scenarios, the goal is to bind the transaction details to the Authorization Response so that integrity is preserved and the User's approval can be proven afterwards (non-repudiation).

The binding mechanism depends on the Credential Format:

- **dc+sd-jwt**: the Wallet binds the transaction data by returning `transaction_data_hashes` (and, when applicable, `transaction_data_hashes_alg`) inside the Key Binding JWT (KB-JWT). See OpenID4VP, Appendix B.3.3 for further details.
- **mso_mdoc**: transaction data is bound through mdoc device authentication. For this format, the Wallet MUST check that the requested transaction data `type` is supported by the document type and authorized by the issuer (KeyAuthorizations). If it is not authorized, the Wallet MUST reject the request due to an unsupported transaction data type. See OpenID4VP, Appendix B.2.1 for further details.

Note

The `state` parameter in an OAuth request is optional, but it is highly recommended. It is primarily used to prevent Cross-Site Request Forgery (CSRF) attacks by including a unique and unpredictable value that the Relying Party can verify upon receiving the response. Additionally, it helps maintain the state between the request and response, such as session information or other data the Relying Party needs after the authorization process.

Note

The `client_metadata` parameter usage is conditional. If `client_id` uses the `x509_hash` prefix, all the Relying Party metadata, other than its public key used for signing the Request Object, MUST be provided in `client_metadata`. However, if it is present and `client_id` uses the `openid_federation` prefix, the Wallet Instance MUST obtain the Relying Party metadata through the OpenID Federation Trust Chain (RPR-96), and MUST NOT use `client_metadata` to override or replace resolved metadata. The only exception is `client_metadata.jwks` (and related encrypted-response capability parameters such as `encrypted_response_enc_values_supported`), which MAY be used exclusively to carry request-specific (ephemeral) public keys for encrypting the Authorization Response in `direct_post.jwt` (see OpenID4VP Section 8.3).

Authorization Response

After obtaining the User authorization and consent for the presentation of the Digital Credentials, the Wallet Instance sends the Authorization Response to the Relying Party `response_uri` endpoint using an HTTP request with the method POST (*WP_091* (page 754)). The response content MUST be encrypted following the high-assurance profile defined in OPENID4VC-HAIP, utilizing response mode `direct_post.jwt` per OpenID4VP Section 8.3. This encryption requires the use of ECDH-ES key agreement on P-256 curve and AES-GCM content encryption (`A128GCM` or `A256GCM`, preferring `A256GCM` when both available), and using the request-specific public key of Relying Party selected from `client_metadata.jwks`, that is identified by its `kid` (*WP_092* (page 754)). The Verifier's public key used to encrypt the Authorization Response is retrieved by the Wallet from the JWKs in the `client_metadata`. According to Section 14.5 of OpenID4VP it is RECOMMENDED the usage of ephemeral keys.

Note

Why the response is encrypted?

The response sent from the Wallet Instance to the Relying Party is encrypted to prevent a malicious agent from gaining access to the plaintext information transmitted within the Relying Party's network. This is only possible if the network environment of the Relying Party employs TLS termination. Such technique employs a termination proxy that acts as an intermediary between the client and the webserver and handles all TLS-related operations. In this manner, the proxy deciphers the transmission's content and either forwards it in plaintext or by negotiates an internal TLS session with the actual webserver's intended target. In the first scenario, any malicious actor within the network segment could intercept the transmitted data and obtain sensitive information, such as an unencrypted response, by sniffing the transmitted data.

Where the following parameters are used (*WP_093* (page 754)):

Name	Description
vp_token	This object MUST contain the presented Digital Credential(s), keyed by the Credential `id` values from the `dcql_query` in the Authorization Request. The `vp_token` MUST be a JSON Object where each key corresponds to a requested Credential id, and each value is either a single presentation or an array of one or more presentations for that Credential. The encoding of each presentation depends on the Credential format, for example: • **dc+sd-jwt**: an SD-JWT VC string (including the appended Key Binding JWT) (*WP_093a* (page 754)). • **mso_mdoc**: a base64url-encoded CBOR `DeviceResponse` corresponding to the requested mdoc presentation (see Open ID4VP Appendix B.2). When multiple mdoc presentations are returned, each MUST be carried in a separate `DeviceResponse` aligned with the corresponding DCQL query item; in this case, the `vp_token` value for that Credential id MUST be an array of `DeviceResponse` values.
state	Unique identifier provided by the Relying Party within the Authorization Request.

Note

Although OpenID4VP considers SD-JWT-based Verifiable Credentials (SD-JWT VC) draft -10, the IT Wallet specification considers SD-JWT VC draft -11 (SD-JWT-VC) to be in line with the version identified in OpenID4VCI.

SD-JWT defines how a Holder can present a Digital Credential to a Relying Party, proving the legitimate possession of the Digital Credential. To do this, the Holder MUST include the `KB-JWT` in the SD-JWT by appending the `KB-JWT` at the end of the SD-JWT (*WP_093b* (page 754)), as represented in the example below

```
<Issuer-Signed-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure N>~<KB-JWT>
```

To validate the signature on the Key Binding JWT, the Relying Party MUST use the key material included in the Issuer-Signed-JWT. The Key Binding JWT (KB-JWT) signature validation MUST use the public key included in the SD-JWT, using the cnf parameter contained in the Issuer-Signed-JWT.

When an SD-JWT is presented, its KB-JWT MUST contain the following parameters in the JWT header (*WP_093c* (page 754)):

Claim	Description
typ	REQUIRED. MUST be `kb+jwt`, which explicitly types the Key Binding JWT as recommended in Section 3.11 of [RFC8725].
alg	REQUIRED. Signature Algorithm using one of the specified in the Section *Cryptographic Algorithms* (page 615).

When an SD-JWT is presented, the KB-JWT signature MUST be verified by the same public key included in the SD-JWT within the *cnf* parameter. The KB-JWT MUST contain the following parameters in the JWT payload:

Claim	Description
iat	REQUIRED. The value of this claim MUST be the time at which the Key Binding JWT was issued, using the syntax defined in [RFC7519].
aud	REQUIRED. The intended receiver of the Key Binding JWT. The value of this parameter MUST match the Relying Party unique entity identifier.
nonce	REQUIRED. Ensures the freshness of the signature. The value type of this claim MUST be a string. The value MUST match with the one provided in the request object.
sd_hash	REQUIRED. The base64url-encoded hash digest over the Issuer-signed JWT and the selected disclosures.
transaction_data_hashes	CONDITIONAL. REQUIRED when the request includes `transaction_data`. Non-empty array of base64url-encoded hashes. Each hash is computed over the exact string value of the corresponding `transaction_data` item.

continues on next page

Table 12.24 – continued from previous page

Claim	Description
transaction_data_hashes_alg	CONDITIONAL. REQUIRED only if the request included `transaction_data_hashes_alg`. String naming the hash algorithm actually used to compute `transaction_data_hashes`; if that parameter was not provided, the hash function MUST be `sha-256`.

Authorization Response Errors

There are cases where the Wallet Instance cannot validate the Request Object or the Request Object results invalid. This error occurs if the Request Object is successfully fetched from the url provided in the parameter `request_uri` but fails the validation checks. This could be due to incorrect signatures, malformed claims, or other validation failures, such as the revocation of the Relying Party.

If the Wallet Instance encounters any such errors during the evaluation of the Authorization Request, it MUST notify the Relying Party by sending an Authorization Error Response (*WP_090* (page 754)). The Wallet Instance sends the Authorization Error Response to the Relying Party `response_uri` endpoint using an HTTP POST request (*WP_090* (page 754)). The Authorization Error Response MUST be encoded in the request body using the format defined by the `application/x-www-form-urlencoded` content type.

Below is a non-normative example of an Authorization Error Response.

```
POST /response_uri HTTP/1.1
HOST: relying-party.example.org
Content-Type: application/x-www-form-urlencoded

state=3be39b69-6ac1-41aa-921b-3e6c07ddcb03&
error=invalid_request&
error_description=...
```

Warning

The current OpenID4VP specification outlines various error responses that a Wallet Instance may return to the Relying Party (Verifier) in case of faulty requests. For privacy enhancement, Wallet Instances SHOULD NOT notify the Relying Party of faulty requests in certain scenarios. This is to prevent any potential misuse of error responses that could lead to gather information that could be exploited.

In the following table are listed error codes and descriptions that are supported for the Authorization Error Response:

Error Code	Description
`invalid_request_uri`	The *request_uri* in the authorization request returns an error, contains invalid data, or is otherwise malformed. **RFC 9101**
`vp_formats_not_supported`	The Wallet Instance does not support any of the vp formats required by the Relying Party. OpenID4VP
`invalid_request_uri_method`	The value of the `request_uri_method` parameter is neither `get` nor `post`. OpenID4VP
`invalid_request`	The request is malformed or inconsistent (e.g., it uses the `vp_token` Response Type but it does not include a `dcql_query` parameter), the Client Identifier Prefix is unsupported, or requirements of a prefix are violated (e.g., `client_id` with the `x509_hash` prefix without the required `client_metadata`). OpenID4VP
`access_denied`	The Wallet did not have the requested credential, the User did not consent, or the Wallet failed to authenticate the User. OpenID4VP
`invalid_client`	The Relying Party's metadata has been resolved based on the Client Identifier (using the `openid_federation` or `x509_hash` prefix), but cannot be authorized due to trust validation failures or is not a valid participant of the federation. OID-FED and OpenID4VP
`invalid_transaction_data`	One or more objects in the `transaction_data` structure are invalid. For instance, those objects contain unknown or unsupported types, malformed (e.g., it is an object of a known type but containing unknown fields or contains fields of the wrong type for the transaction data type) or missing fields, invalid values (e.g., the `credential_ids` does not match), or references to unavailable Credentials. OpenID4VP

Relying Party Response

As defined in Section 8.2. (Response Mode `direct_post`) of the OpenID4VP specification, if the Response URI has successfully processed the Authorization Response or Authorization Error Response, it MUST respond with an HTTP status code of 200 with `Content-Type` of `application/json` and a JSON object in the response body.

In the **Same Device Flow**, the Relying Party MUST add the `redirect_uri` parameter to the JSON object in the response body. Upon receiving the `redirect_uri`, the Wallet Instance MUST perform a redirect to the URL specified by the `redirect_uri`. This redirect allows the Relying Party to seamlessly resume interaction with the User on the device which initiated the flow, after the Wallet Instance has transmitted the Authorization Response to the designated `response_uri`.

The Relying Party MUST include a response code within the `redirect_uri`. The response code is a fresh, cryptographically random number used to ensure only the receiver of the redirect can fetch and process the Authorization Response. The number could be added as a path component, as a parameter or as a fragment to the URL. It is RECOMMENDED to use a cryptographic

random value of 128 bits or more at the time of the writing of this specification. Even if an adversary manages to steal the random value used in the request to the status endpoint, their user-agent would be rejected due to the missing cookie in the request.

Warning

For security reasons and to prevent endpoint mix-up attacks, the value contained in the `redirect_uri` parameter MUST be one of those attested by a trusted third party, such as those provided in the `openid_credential_verifier` metadata within the `redirect_uris` parameter, obtained from the Trust Chain about the Relying Party (*WP_094a* (page 754)).

Relying Party Response Errors

If any validation check, performed by the Relying Party on the Authorization Response from the Wallet Instance, fails; the Response URI endpoint MUST return an error response. The structure of this error response should be determined by the specific nature of the error encountered. The response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`400 Bad Request`	`invalid_request`	The response cannot be processed because it is missing required parameters, contains invalid parameters or is otherwise malformed.
`400 Bad Request`	`invalid_request`	The Credentials presented are malformed, invalid or revoked.
`400 Bad Request`	`invalid_request`	The credential presentation, contained in the `vp_token` object, is malformed, doesn't have the required parameters or is incorrectly formatted.
`400 Bad Request`	`invalid_request`	The "sd-jwt" returned is malformed, missing required parameters or incorrectly formatted.
`403 Forbidden`	`invalid_request`	The signature of the KB-JWT is invalid or does not match the associated public key (JWK) referenced in the Issuer signed SD-JWT.
`403 Forbidden`	`invalid_request`	The nonce value provided is incorrect or otherwise malformed.
`403 Forbidden`	`invalid_request`	Trust could not be established with the Credential Issuer.

continues on next page

Table 12.26 – continued from previous page

Status Code	Error Code	Description
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Response URI Endpoint encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Response URI Endpoint is temporarily unavailable (e.g., due to maintenance or overload).

Below there are two examples of HTTP responses using `application/json` that include both the `error` and `error_description` members:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "Trust cannot be established with the issuer: https://issuer.example.com
"
}
```

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The vp_token is malformed, missing required parameters or incorrectly f
ormatted"
}
```

Status Endpoint

This specification introduces the Relying Party Status Endpoint for implementations that choose to use it. This endpoint is an internal security feature of the implementation and is not required for interoperability.

Whether the flow is Same Device or Cross Device, the user-agent needs to check the session status at the endpoint made available by the Relying Party (status endpoint). This check MAY be implemented in the form of JavaScript code, within the page that shows the QRCode or the href button pointing to the request URL. The JavaScript code makes the user-agent check the status endpoint using either a polling strategy (in seconds) or a push strategy (e.g., WebSocket).

Since the HTML page and the status endpoint are implemented by the Relying Party, the implementation details of this solution are the responsibility of the Relying Party, as this is related to the Relying Party's internal API. However, the text below describes an example implementation.

The Relying Party binds the request of the user-agent, with a session cookie marked as `Secure` and `HttpOnly`, with the issued request. The request url SHOULD include a parameter with a random value. The HTTP response returned by this status endpoint MAY contain the HTTP status codes listed below:

- **201 Created**. The signed Request Object was issued by the Relying Party that waits to be downloaded by the Wallet Instance at the `request_uri` endpoint.
- **202 Accepted**. This response is given when the signed Request Object was obtained by the Wallet Instance.
- **200 OK**. The Wallet Instance has provided the presentation to the Relying Party's `response_uri` endpoint and the User authentication is successful. The Relying Party updates the session cookie allowing the user-agent to access to the protected resource. A redirect URL is provided carrying the location where the user-agent is intended to navigate.

Status Endpoint Errors

If instead any validation check performed by the Relying Party fails, the QRCode page SHOULD be updated with an error message. Moreover, the status endpoint MUST return an error response, whose structure depends on the nature of the error. The response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`401 Unauthorized`	`authentication_failed`	The Wallet Instance or its User have rejected the request, the request is expired, or other errors prevented the authentication.
`403 Forbidden`	`invalid_session`	Either the session id provided in the request is invalid.

Redirect URI

The `redirect_uri` value MUST be used with an HTTP method GET by the user-agent to redirect the User to a specific Relying Party's endpoint in order to complete the process.

Redirect URI Errors

When the user-agent is redirected to the Redirect URI provided by the Relying Party, several errors may occur that prevent the successful completion of the process. These errors are critical as they directly impact the User experience by hindering the seamless flow of information between the Wallet Instance and the Relying Party. Handling these errors requires clear communication to the User within the returned navigation web page. Relying Party MUST implement the error handling and validation mechanisms for Redirect URIs defined in this specification. Below are potential errors related to the Redirect URI, the error response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`403 Forbidden`	`invalid_request`	The Redirect URI provided by the Relying Party does not match any of the URIs linked with the User session. (**RFC 6749#section-4.1.2.1**)
`403 Forbidden`	`invalid_request`	The User session is invalid or expired.
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled due to an internal server error. (**RFC 6749#section-4.1.2.1**).
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the service is temporarily unavailable (e.g., due to maintenance or overload). (**RFC 6749#section-4.1.2.1**).

12.2.2 Proximity Flow

This section describes how a Relying Party Instance requests the presentation of an *mdoc-CBOR* Credential to a Wallet Instance as specified in the *ISO 18013-5 Specification*.

The high-level presentation phase is structured into three broad sub-phases as depicted in the following figure:

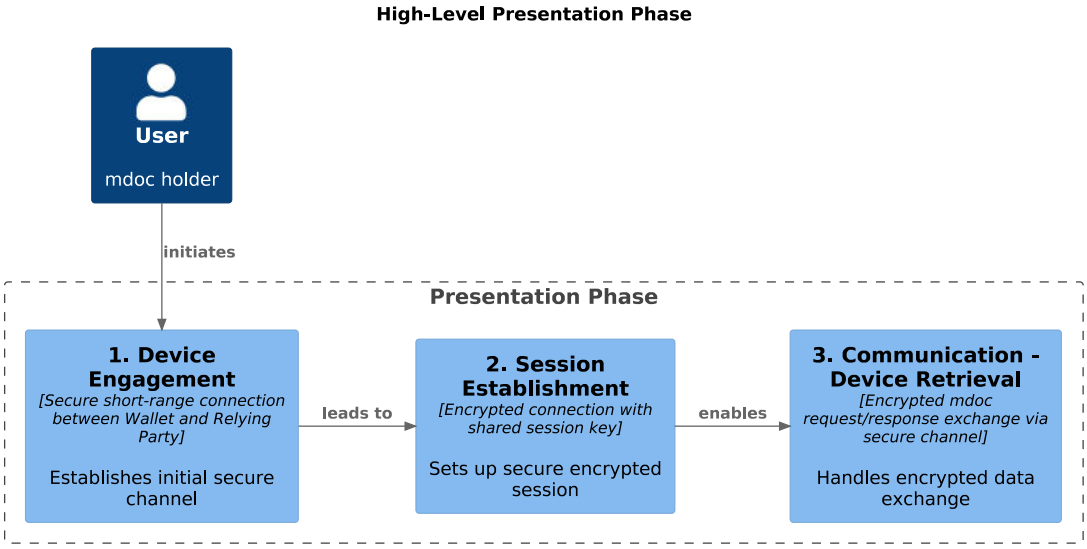


Fig. 12.14: High-Level Presentation Flow in proximity.

The sub-phases are described below:

1. **Device Engagement**: This subphase begins when the User is prompted to disclose specific attributes from the mdoc(s). The objective of this subphase is to establish a secure communication channel between the Wallet Instance and the Relying Party Instance, so that the mdoc requests and responses can be exchanged during the communication subphase. The messages exchanged in this subphase are transmitted through short-range technologies to limit the possibility of interception and eavesdropping. Device Engagement data may be transferred using either QR code or NFC technologies.

2. **Session Establishment**: During the session establishment phase, the Relying Party Instance sets up a secure connection. All data transmitted over this connection is encrypted using a session key, which is known to both the Wallet Instance and the Relying Party Instance at this stage. The established session MAY be later terminated based on the conditions as detailed in [ISO18013-5 #12.2.4].

3. **Communication - Device Retrieval**: The Relying Party Instance encrypts the mdoc request with the appropriate session key and sends it to the Wallet Instance together with its public key in a session establishment message. The Wallet Instance uses the data from the session establishment message to derive the session key and

then to decrypt the mdoc request. During the communication subphase, the Relying Party Instance has the option to request information from the Wallet Instance using mdoc requests and responses. The primary mode of communication is the secure channel established during the session setup. The Wallet Instance encrypts the mdoc response using the session key and transmits it to the mobile Relying Party Instance via a session data message.

Relying Party and Wallet Instances registered in the IT-Wallet ecosystem MUST support at least the following flows:

- *Supervised Device Retrieval flow* where a human Relying Party is overseeing the verification process in person, or *unsupervised flow* where verification might happen through automated systems without human oversight (*WP_095* (page 754)).
- *Relying Party Instance Authentication* following the mechanisms defined in the ISO18013-5 for the *reader authentication* (*WP_098* (page 754)).
- Domestic *Document Type* and *Namespaces* defined in this technical specification in addition to those already defined in the ISO18013-5 for the mDL (see *mdoc-CBOR Credential Format* (page 366) for more details) (*WP_099* (page 754)).

The following table shows the supported Device Engagement technologies (*WP_097* (page 754)), specifying which are mandatory.

Transmission technology	ISO 18013-5 Wallet Instance	ISO 18013-5 RP Instance	IT Wallet Wallet Instance	IT Wallet RP Instance
QR code	C[a]	M	MUST	C – MUST if the device is equipped with a camera or QR code reader and BLE.
NFC	C[a]	M	RECOMMENDED	C – MUST if the device is equipped with an NFC reader.

Key: C = Conditional | M = Mandatory | [a]Support for at least one of these methods is mandatory (*WP_097a* (page 754))

The following table shows the supported Device Retrieval technologies, specifying which are mandatory.

Transmission technology	ISO 18013-5 Wallet Instance	ISO 18013-5 RP Instance	IT Wallet Wallet Instance	IT Wallet RP Instance
BLE	C[a]	M	MUST	C – MUST if the device is equipped with a camera or QR code reader and BLE.
NFC	C[a]	M	RECOMMENDED	C – MUST if the device is equipped with an NFC reader.

Key: C = Conditional | M = Mandatory | [a]Support for at least one of these methods is mandatory (*WP_096b* (page 754))

Note

From the second edition, version 3, ISO18013-5 does not define or support Server Retrieval as a transport option. Only proximity retrieval methods (NFC, BLE, and optionally Wi-Fi Aware) are specified (*WP_096* (page 754)). Therefore, Server Retrieval is not considered in this flow (*WP_096a* (page 754) and PPR-023).

The following figure illustrates the low-level flow compliant with ISO 18013-5 for proximity flow.

Step 1: The User opens the Wallet Instance initiating the process.

Step 2: The User authenticates itself to the Wallet Instance. This can be done by the Wallet Instance or a Wallet Secure Cryptographic Application (WSCA). It is a prerequisite for accessing sensitive data and presenting attributes (*WP_100* (page 754)).

Step 3: The User selects the proximity presentation functionality.

Step 4: [Optional] If the initial authentication in Step 2 was not done through WSCA, a separate authentication via WSCA MAY be required (*WP_100* (page 754)).

Step 5: The Wallet Instance generates a new ephemeral Elliptic Curve key pair for secure communication (*WP_101* (page 754)). The public key (`EDeviceKey.Pub`) will be exchanged with the Relying Party Instance to derive a shared session key, which is then used for session encryption. This is part of the device engagement process.

Box A

The Wallet Instance and Relying Party Instance exchange *Device Engagement* data via QR code or via NFC Connection Handover (*WP_097* (page 754)).

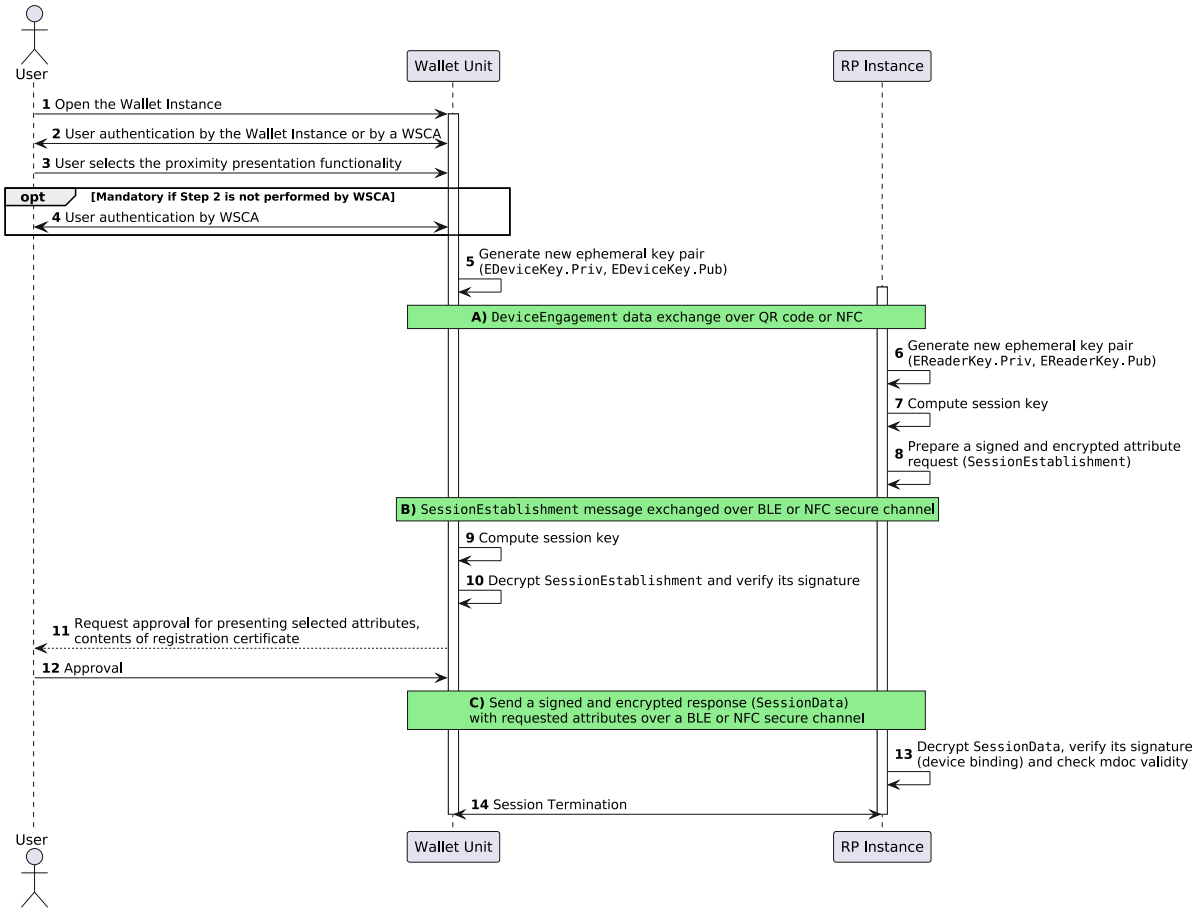


Fig. 12.15: Low-Level Presentation Flow in proximity.

Refer to:

- Sec 8.2.2.1 for `DeviceEngagement` over QR code
- Sec 8.2.2.2 for `DeviceEngagement` over NFC

Step 6: The Relying Party Instance generates its ephemeral key pair (`EReaderKey.Priv`, `EReaderKey.Pub`). The private key (`EReaderKey.Priv`) MUST be kept secret, and the public key (`EReaderKey.Pub`) MUST be used in establishing the session.

Step 7: The Wallet Instance and Relying Party Instance independently MUST derive the session keys using their private ephemeral key and the other party's public ephemeral key through a key agreement protocol. This ensures session encryption. In this particular step, the Relying Party Instance MUST compute its session key (PPR-002 and *WP_104* (page 754)).

Step 8: The Relying Party Instance MUST prepare a `SessionEstablishment` message. This message MUST be signed by the Relying Party Instance (mdoc reader authentication as specified in [ISO18013-5 #12.5]) and encrypted using the session keys derived in the previous step. The `SessionEstablishment` message MUST include the `EReaderKey.Pub` and a request for specific attribute(s) (PPR-002).

Below is a non-normative example using the diagnostic notation of a CBOR-encoded `SessionEstablishment` message that contains an mdoc request for an mDL Digital Credential.

```
{
  "version": "1.0",  % Version
  "docRequests":
  [
    {
    "itemsRequest":
    24(<<  % Embedded CBOR data item
      {
        "docType": "org.iso.18013.5.1.mDL",
        "nameSpaces":
        {
          "org.iso.18013.5.1":
          {
            "family_name": true,
            "document_number": true,
            "driving_privileges": true,
            "issue_date": true,
            "expiry_date": true,
            "portrait": false
          }
        }
      }
    >>)
```

```
    }
  ],
  "readerAuthAll":
  [
    h'a10126', % COSE_Sign1 authentication header
    {
      33: h'308201253081cda00302010202012a300a06082a8648ce3d0403023020311e301c06035504030c1553
      6f6d6520526561646572204175746861726974792079301e170d323331313234313032383232325a170d3238313132
      323130323832325a301a311830160603550403c0f536f6d65205265616465722046b65793059301306072a86
      48ce3d020106082a8648ce3d03010703420004aa1092fb59e26ddd182cfdbc85f1aa8217a4f0fae6a6a5536b
      57c5ef7be2fb6d0dfd319839e6c24d087cd26499ec4f87c8c766200ba4c6218c74de50cd1243b'
    },
    null, % No additional reader authentication
    h'58a0d421a7e53b7db0412a196fea50ca6d4c8a530a47dd84d88588ab145374bd0ab2a724cf2ed2facf32c718
    4591c5969efd53f5aba63194105440bc1904e1b9'  % Reader authentication
    signature
  ]
}
```

Box B

The Relying Party Instance MUST transmit the encrypted and signed `SessionEstablishment` message to the Wallet Instance over an NFC or a BLE secure connection that was established based on the Device Engagement information. Refer to:

- Sec 8.2.2.3 for `SessionEstablishment` over BLE, and
- Sec 8.2.2.5 for `SessionEstablishment` over NFC

Step 9: The Wallet Instance MUST compute the session key, as described in Step 7.

Step 10: Upon receiving the `SessionEstablishment` message, the Wallet Instance MUST decrypt it using the shared session key and MUST verify the Relying Party Instance's signature (mdoc reader authentication as specified in [ISO18013-5 #12.5]) to ensure its authenticity (PPR-002 and *WP_105–106* (page 754)).

Step 11: The Wallet Instance MUST decrypt the attribute request and MUST prompt the User for their consent to release the requested attributes (*WP_107* (page 754)). It MUST also display the contents of the Relying Party's Registration Certificate to ensure transparency about the requested attributes and its registered purpose (*WP_107a* (page 754)).

Step 12: The User reviews the request and the Relying Party's registration information and then approves the presentation of the requested attributes.

Box C

After receiving User approval, the Wallet Instance MUST retrieve the requested mdoc Digital Credentials (PPR-006 and *WP_108* (page 754)). It then MUST prepare a `SessionData` message containing these Digital Credentials, and it MUST sign the required authentication data (as part of the mdoc authentication process, as specified in [ISO18013-5 #12.4]) as per (*WP_109–110* (page 754)). It MUST encrypt it using the established session keys before transmitting it to the Relying Party Instance over the secure channel (*WP_111* (page 754)). The signing ensures device binding and data integrity. The mdoc response MUST be encoded in CBOR, with its structure outlined in [ISO18013-5 #10.3] (PPR-029, PPR-030, and *WP_112* (page 754)). Refer to (*WP_112a–112b* (page 754)):

- Sec 8.2.2.4 for `SessionData` over BLE, and
- Sec 8.2.2.5 for `SessionData` over NFC

Below is a non-normative example using the diagnostic notation of a CBOR-encoded `SessionData` that contains the mdoc response of an mDL Digital Credential.

```
{
    "version": "1.0",
    "documents":
    [
      {
        "docType": "org.iso.18013.5.1.mDL",
        "issuerSigned":
        {
          "nameSpaces":
          {
            "org.iso.18013.5.1":
            [
              24(<<
                {
                    "digestID": 0,
                    "random":
                    h'790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a96406',
                    "elementIdentifier": "family_name",
                    "elementValue": "Rossi"
                }
              >>),
              24(<<
                {
                    "digestID": 3,
                    "random":
                    h'0c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f',
                    "elementIdentifier": "issue_date",
```

```
      "elementValue": 1004(2025-03-27)
    }
  >>),
  24(<<
    {
      "digestID": 4,
      "random":
      h'0a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e3339670203178e',
      "elementIdentifier": "expiry_date",
      "elementValue": 1004(2030-03-27)
    }
  >>),
  24(<<
    {
      "digestID": 8,
      "random":
      h'88e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6c97cf214',
      "elementIdentifier": "document_number",
      "elementValue": "XX1234567"
    }
  >>),
  24(<<
    {
      "digestID": 9,
      "random":
      h'944758b43602b01ad68911b062349845492c04c6a78129bcf8cb5fb1396af2fc',
      "elementIdentifier": "portrait",
      "elementValue":
      h'ffd8ffe000104a464946000101010090009000 0ffdb004300130d0e110e0c
      13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241
      435f82
      606871757b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f
      767676
      7676767676767676767676767676767676767676767676767676767676767676767676
      767676
      76767676767676ffc0001108001800640301220002110103110 1ffc4001b00000301000301
      000000
      0000000000000000050604010203 07ffc400321000010303030205020309000000000000001
      020304
      0005110612211331141551617122410781a1163542527391b2c1f1ffc40015010101000000
      000000
      0000000000000000001ffc4001a1101010100030100000000000000000000000141112131
      61ffda
      000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e770
      9723a1
      d0dae146ddfbb3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb
      95ff00
```

```
        eb9ff14d215de66af089ce44b7dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65
        d6b95a
        147381ea0d495b933275fe6bba75c114104a8ba410413e983dff004f5af5d34b4b4cde632d
        0bf1fd
        1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e856001ebc03c7ce29dd9eef1e
        f10fc4
        47dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e54528750946e
        c67048
        50cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493
        f602a0
        75559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7
        290875
        db4be88ab384bbbbbfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30
        ab420a
        7a6d3240c718cf031fa9ef4c9ad550205aa02951df4a1d6c8421b015b769db8c9229837ea2
        be8b1b
        0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2edbd4584e7af9c64e3f96b9beb28d4
        ac4093
        1e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4a179c1c1e460938f
        9149ef
        655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dce1
        5eed9b
        71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c52
        56a79e
        706d52d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344
        e847dc
        ac9dcb5abba8d145061d43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a1
        5f71dd
        27d451453e09d1a21450961cbb3ea8a956433b781f1ce33dfed54f0e2b50a2b71d84ed6db1
        8028a2
        8175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a66b726e33de9ea6f1b52b181c7
        60724e
        47b514520a5a28a283ffd9'
    }
  >>),
  24(<<
    {
        "digestID": 10,
        "random":
        h'577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f',
        "elementIdentifier": "driving_privileges",
        "elementValue": [
          {
            "vehicle_category_code": "A",
            "issue_date": 1004("2020-09-17"),
            "expiry_date": 1004("2031-06-10")
            }
```

```
            ]
        }
      >>)
    ]
  },
  "issuerAuth":
  [
    << {1: -7} >>,
    {
      33: h'30820208308201afa00302010202142eb39c647c81836bcf79fa9cd0b201ec0bf52307300a0
      6082a8648ce3d0403023064310b3009060355040613025553311330110603550 4080c0a43616c6966
      6f726e6961131630140603550 4070c0d53616e204672616e636973636f3113301106035504 0a0c0a4
      d7920436f6d70616e7931133011060355040 30c0a6d79736974652e636f6d301e170d323530333237
      3135353532305a170d3235303430363135353532305a3064310b3009060355040613025553311330 1
      106035504080c0a43616c69666f726e6961131630140603550 4070c0d53616e204672616e63697363
      6f31133011060355040a0c0a4d7920436f6d70616e7931133011060355040 30c0a6d79736974652e6
      36f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004f33da72d0dd0009b62221b
      0e839099b12dab5e01021124ebf9060422e648f3c3ec6614a86da1e91e552b2ae35e04d3058ae82b5
      c65a7f1f26800cb4499652a09a33f303d303b0603551d11043430328630 68747470733a2f2f637265
      64656e7469616c2d6973737565722e6f6964632d6665646572617469 6f6e2e6f6e6c696e65300a060
      82a8648ce3d040302034700304402204d1f0819971652b79ebe4825547de3d5554d2f41410225e6b1
      3dab949cda125e022079ba71b823619e49719dce5daa565bf745d3d97e2b87c7f7d6a626f981e653
      ed'
    },
    <<
      24(<<
        {
          "version": "1.0",
          "digestAlgorithm": "SHA-256",
          "docType": "org.iso.18013.5.1.mDL",
          "valueDigests": {
            "org.iso.18013.5.1":
            {
                0: h'f46b65d5060ad060ab9be62ff22ea8633437619ebdc7fa81f2d151159e92bffe',
                1: h'e506545f6a6fd5d982670b4d62fc2b0688dc8f26754e7b0c574d63f5d72a85ac',
                2: h'cfcf96fa12d100eeed5f00183d3b6a0888baa47eae85b5b95037eca7bbc0d07e',
                3: h'8b0772252b0e06b611676b6b3402eb33bf866eb145e49f4d5f23215e6a047772',
                4: h'14135c96693e2ab08d956876ee491357d906a6dd125557196dfb9811ba54aa8d',
                5: h'86dcbd99233fbb84a9a2dce3a864a425e6e809300067a4475e3ea2a4d233dc74',
                6: h'2e9512d35ea225e69e7b2180ecc1678dcc3e77a16e36427e64b4f0e2861b4d3a',
                7: h'4efe55c36f6249d23c473a125afc5181aa3063393649478155497 1b72ff13700',
                8: h'cc44a4f9983c5b0b1efc0e82e2867c8d5bbdf89c34bff16a1953c923bb4e4b3e',
                9: h'775eb2af0aa55f2071d62662b35c99698ae3bc0e2c4af5724ff88476cddd152f',
               10: h'915d0ad53dd23dace34968c263d307c04701a9bb9dc9865af91dc409786fd833',
               11: h'47d89ff4fb513044e6f2394236755ac0abf3e4f4a46f40454a458a59f8b7a6fb'
            },
            "org.iso.18013.5.1.IT": {
```

```
                12: h'd665c50c4364c7cbf4ab9461b9bbb228f37ad278b9fb61283550951624d4d9ae',
                13: h'dc9d032a64866e33d06f48a882989b5747da3638f0d216a2275191ed3395fdec'
               }
            },
            "deviceKeyInfo": {
              "deviceKey": {
                 1: 2,
                -1: 1,
                -2: h'f96b29873b61f05403e2963a7ecbc799c9aab28d8a6629e5848cfdef85442866',
                -3: h'a9fef033a900c63e3894d8deb805a2a1fb55ef0d2b88e3c0d3336408186485ef'
              }
            },
            "validityInfo": {
              "signed": 0("2025-03-27T00:00:00Z"),
              "validFrom": 0("2025-03-27T00:00:00Z"),
              "validUntil": 0("2026-03-27T00:00:00Z")
            },
            "status": {
              "status_list": {
                "idx": 1340,
                "uri": "https://statusprovider.example.org/statuslists/1"
              }
            }
          }
        >>)
      >>,
      h'd09f9acdf7a6be5e4aeb405bfb3b297b1b8003bcf52558a2f39fc6e5cffed40f18f49d2cc0e72a2a
      5645
      8d8aade591dee8d6540e639bca637f94bd9fa56f345c'
    ]
  },
  "deviceSigned":
  {
    "nameSpaces": 24(<< {} >>),
    "deviceAuth":
    {
      "deviceSignature":
      [
        << {1: -7} >>,
        {},
        null,
        h'E99521A85AD75943BF5AEAE68943BF5AE249943BF5AE943BF5AE43BF5AEDE99521A85AD75943BF
        5AEAE
        68943BF5AE249943BF5AE943BF5AE43BF5AED'

      ]
    }
```

```
        }
      }
    ],
    "status": 0
}
```

Step 13: The Relying Party Instance receives the `SessionData`, then it MUST decrypt it, and it MUST verify the Wallet Instance's signature to ensure the data's integrity and that it originates from the expected device (device binding). It also MUST check the validity of the mdoc, including its Issuer's signature. In case of long-lived Digital Credentials, it SHOULD also check the revocation status using the TOKEN-STATUS-LIST.

Step 14: Once the data exchange is complete, either party can terminate the session. The session can be terminated by sending the status code for session termination in a `SessionData` message; this can be sent together with an mdoc request or response [ISO18013-5 #12.2.4] (*WP_113c* (page 754)). If BLE is used, this can involve sending a status code for session termination or the "End" command. In this scenario, the GATT Client (Relying Party Instance) MUST unsubscribe from characteristics and disconnect from the GATT server (Wallet Instance) (PPR-007, *WP_113b* (page 754), and *WP_114* (page 754)).

Final Consideration: The presentation flow focused on the technical data exchange in proximity settings. It is crucial to recognise that supervised proximity flows involving a human verifier play a vital role in many use cases (e.g., age verification at a store, identity check by law enforcement). The human element adds a layer of identity verification through visual inspection and comparison, contributing to User Binding and overall authentication assurance aspects not fully captured in a purely technical presentation flow.

Note

During each credential presentation transaction executed through the proximity flow, the Wallet Instance MUST create and maintain a corresponding transaction record in the transaction log (see *Wallet Instance Dashboard and Transaction Logging* (page 299)).

The transaction record MUST be created once the Wallet Instance has successfully established the session and accepted the reader for processing (i.e., after decrypting the `SessionEstablishment` message and verifying reader authentication, Step 10). At this point, the record MUST include the transaction metadata and the request context available at that stage (e.g., requested attestation type(s) such as `docType`, and the identifier(s) of the requested attributes), without logging any attribute values.

The record MUST be updated as the transaction progresses to reflect the evolving transaction state and result context (e.g., what was actually presented after User consent and response preparation/sending, Steps 11–12), without logging any attribute values.

The record MUST be finalized when the transaction ends, indicating the outcome (e.g., completed, failed, or session terminated; Steps 13–14 and Session Termination).

`DeviceEngagement` over QR Code

The following figure illustrates the low-level flow compliant with ISO 18013-5 for `DeviceEngagement` over QR Code corresponding to Box A in Figure *Low-Level Presentation Flow in proximity.* (page 514).

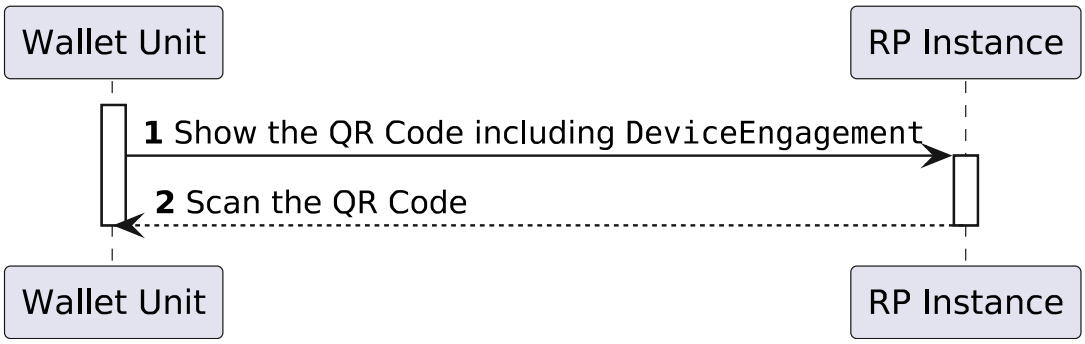


Fig. 12.16: Device Engagement over QR Code.

Step 1: The Wallet Instance presents a QR Code to the Relying Party Instance. The QR code MUST contain a URI with "mdoc:" as scheme and the `DeviceEngagement` structure specified in Section 9.1 encoded using base64url-without-padding, according to RFC 4648, as path (*WP_102a* (page 754)).

Non-Normative Example with BLE as Data Retrieval

Below is a non-normative example using the diagnostic notation of a CBOR-encoded `DeviceEngagement` that utilizes QR for Device Engagement and Bluetooth Low Energy (BLE) for data retrieval.

```
{
  0: "1.1", % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
```

```
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
  ],
  2: % DeviceRetrievalMethods (Device engagement using QR code with BLE for retrieval)
  [
    [
      2, % BLE
      1, % Version
      { % BLE options
        0: false, % no support for mdoc peripheral server mode
        1: true,  % support for mdoc central client mode
        11: h'45EFEF742B2C4837A9A3B0E1D05A6917' % UUID of mdoc client central mode
      }
    ]
  ],
  5: % OriginInfos (Required because Capabilities is present)
  [],
  6: % Capabilities (Defines supported features)
  {
    2: false, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
    3: true % ReaderAuthAllSupport (Supports reader authentication)
  }
}
```

Non-Normative Example with NFC as Data Retrieval

Below is a non-normative example using the diagnostic notation of a CBOR-encoded `Device Engagement` that utilizes QR for Device Engagement and NFC for data retrieval.

```
{
  0: ”1.1”, % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
```

```
    ],
    2: % DeviceRetrievalMethods (Device engagement using QR code with BLE for retrieval)
    [
      [
        1, % NFC
        1, % Version
        { % NFC options
          0: uint, % Maximum length of command data field
          1: uint, % Maximum length of response data field
        }
      ]
    ],
    5: % OriginInfos (Required because Capabilities is present)
    [],
    6: % Capabilities (Defines supported features)
    {
      2: false, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
      3: true % ReaderAuthAllSupport (Supports reader authentication)
    }
  }
```

Step 2: The Relying Party uses its Relying Party Instance to scan the QR code and retrieve the `DeviceEngagement` data from the mdoc. It MUST select one of the transmission technologies from the ones provided in the `DeviceEngagement` structure.

`DeviceEngagement` over NFC

The following figure illustrates the low-level flow compliant with ISO 18013-5 for `DeviceEngagement` over NFC corresponding to Box A in Figure *Low-Level Presentation Flow in proximity.* (page 514) (*WP_103* (page 754)).

`DeviceEngagement` over NFC is based on the NFC Forum Connection Handover Technical Specification, Version 1.5. Only Reader/ Writer mode using a Type 4 Tag is supported. The Connection Handover protocol is always initiated by the Relying Party Instance, which makes the role of Handover Requester. The Wallet Instance MUST use either Static Handover or Negotiated Handover: - **Static Handover**: The Relying Party Instance retrieves a Handover Select message directly from the Wallet Instance's Type 4 Tag. This message contains at least one Alternative Carrier Record, each indicating a retrieval method supported by the Wallet Instance (*WP_103a* (page 754)). The Relying Party Instance MUST select one of these transmission technologies. (see Step 1) - **Negotiated Handover**: The Wallet Instance includes the service `urn:nfc:sn:handover` in a Service Parameter Record of the initial NDEF (NFC Data Exchange Format) message (*WP_103b* (page 754)). Upon selecting this service, the Relying Party Instance sends a Handover Request with an Alternative Carrier Record for each carrier it supports. The Wallet

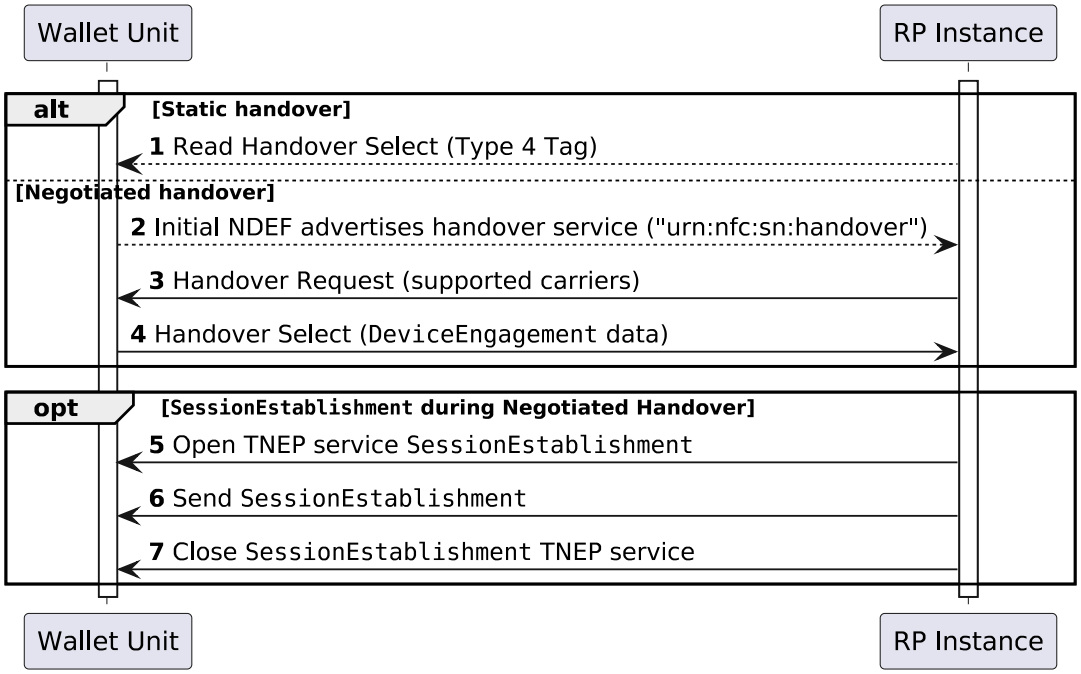


Fig. 12.17: Device Engagement over NFC.

Instance replies with a Handover Select message containing exactly one selected carrier. (See Step 2-4)

Step 1: The Relying Party Instance reads the Wallet's NFC Type 4 Tag to obtain a Handover Select message, which includes: - Alternative Carrier Record is an NDEF record inside a Handover Select or Handover Request message. It points to one possible communication technology (called a "carrier"), such as NFC or BLE. It tells the reader about the supported carrier and a pointer (Auxiliary Data Reference) to more detailed information. The Alternative Carrier Record for the NFC device retrieval transmission technology MUST reference the Carrier Configuration Record with the ID reference "nfc". - Carrier Configuration Record provides the technical parameters needed actually to use that carrier. For NFC device retrieval transmission technology, it MUST have the type "iso.org:18013:nfc" and the ID reference "nfc". The binary content of the Carrier Configuration Record MUST be encoded according to Table 1 of [ISO18013-5 #9.2.2] (*WP_103d* (page 754)).

For example: For NFC, this defines maximum APDU (Application Protocol Data Unit) command/response lengths; For BLE, it defines the Wallet Instance service UUID, characteristic UUIDs, MTU (Maximum Transmission Unit) size, and optional connection parameters; If early `SessionEstablishment` is supported, it also lists the TNEP (Tag NDEF Exchange Protocol) service name used to send the `SessionEstablishment` message during handover.

Note

For the NFC device retrieval transmission technology, the contents of the Alternative Carrier Record and Carrier Configuration Record(s) MUST comply with [ISO18013-5 #9.2.2]. For the BLE device retrieval transmission technology, the contents of the Alternative Carrier Record and Carrier Configuration Record(s) MUST comply with [ISO18013-5 #11.1.2].

- Auxiliary Data Record MUST carry the DeviceEngagement structure from the Wallet Instance to the Relying Party Instance as part of the auxiliary NDEF record in the Handover Select message. This record has the type `iso.org:18013:deviceengagement`, the ID reference "mdoc", and uses the NFC forum external type format (`0x04`). For each Alternative Carrier record, the Auxiliary Data Reference MUST point to the NDEF record containing the `DeviceEngagement` Structure (*WP_103e* (page 754)).

Step 2: The Relying Party Instance reads the Wallet Instance's Initial NDEF (NFC Data Exchange Format) message, which contains a service parameter record for `urn:nfc:sn:handover`, indicating the Wallet supports Negotiated Handover.

Step 3: The Relying Party Instance sends a Handover Request to the Wallet Instance listing the supported carriers.

Step 4: The Wallet Instance returns Handover Select constructed in response to the received Handover Request message. The contents of the Handover Select message is the same as Step 1 (*WP_103f* (page 754)).

Note

Use of Negotiated Handover for Device Engagement allows negotiation of transfer methods. For BLE, it additionally allows negotiation of keys used by the transmission layer. This provides improved user experience and enhances the security of data transmission [ISO18013-5 #9.2.1].

Note

Proceed only if the `DeviceEngagement` Capabilities include `HandoverSessionEstablishmentSupport` set to `true` (*WP_103c* (page 754)). Otherwise, skip the early `SessionEstablishment`. The early `SessionEstablishment` is sent via a dedicated TNEP service; the same `SessionEstablishment` MUST also be sent again during data retrieval and MUST match. If it does not match, the Wallet Instance terminates (*WP_103g* (page 754)). If the early `SessionEstablishment` fails to send, proceed as normal (*WP_103h* (page 754)).

Step 5: [Optional] Relying Party Instance opens the TNEP service named [urn:placeholder] with the Wallet Instance during the negotiated handover to deliver the early `SessionEstablishment` message.

Step 6: Relying Party Instance sends `SessionEstablishment` (e.g., `EReaderKey` + encrypted `DeviceRequest`). Wallet Instance processes it; data retrieval has not started yet.

Step 7: Relying Party Instance closes the TNEP service.

Note

If an optional `SessionEstablishment` message is sent during Negotiated Handover (Step 5), the Wallet Instance MUST verify that it matches the `SessionEstablishment` message received during Device Retrieval (using BLE or NFC secure channel). This verification is required to ensure a correct Session Binding.

Non-Normative Example

Below is a non-normative example of a `DeviceEngagement` structure for Data Retrieval over BLE and NFC.

```
{
  0: "1.1", % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
  ],
  2: % DeviceRetrievalMethods (Absent when using NFC for device engagement)
  [],
  5: % OriginInfos (Required because Capabilities is present)
  [],
  6: % Capabilities (Defines supported features)
  {
    2: true, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
    3: true % ReaderAuthAllSupport (Supports reader authentication)
```

```
    }
  }
```

`SessionEstablishment` over BLE

The following figure illustrates the low-level flow compliant with ISO18013-5 for `SessionEstablishment` over BLE corresponding to Boxes B in Figure 8.10.

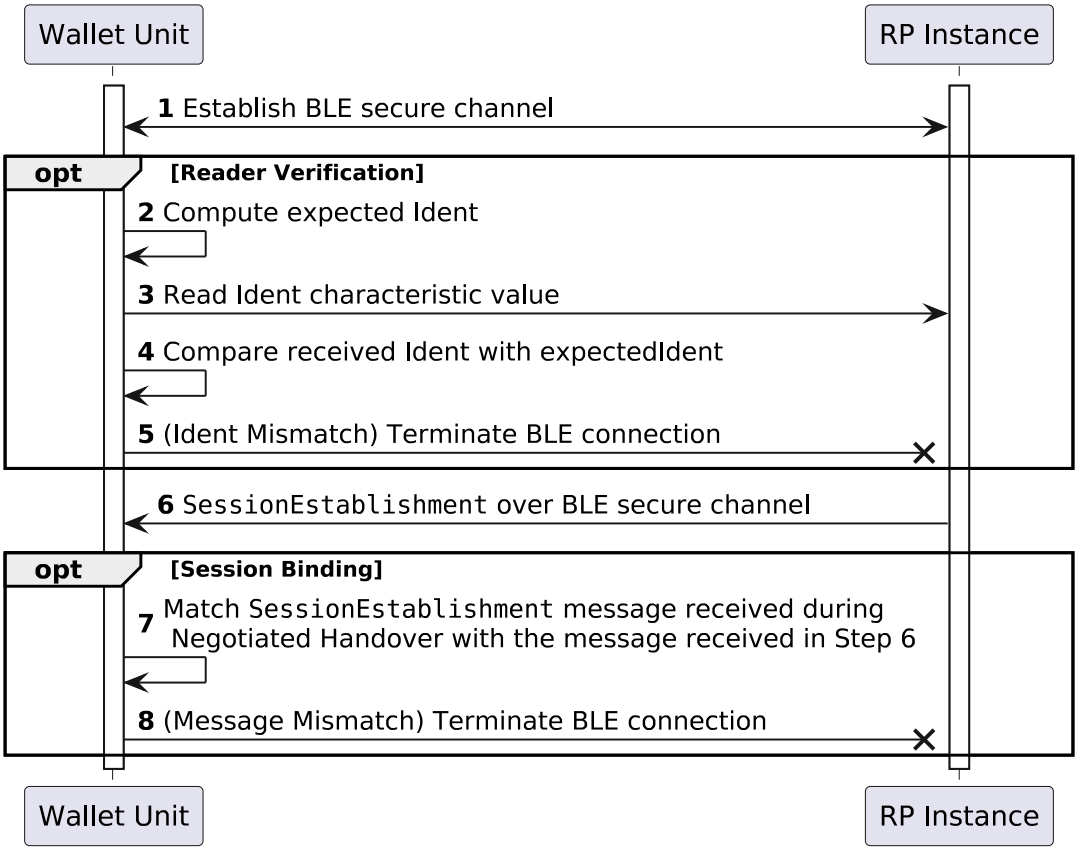


Fig. 12.18: Session Establishment over BLE.

Step 1: The Wallet Instance and Relying Party Instance establish a secure BLE connection [ISO18013-5 #11.1]. The Relying Party Instance (central) connects to the Wallet Instance (peripheral) using the Wallet Instance service UUID provided by DeviceEngagement, then discovers services/characteristics, and enables notifications as needed (*WP_112c* (page 754)).

Step 2-5: [Optional] Wallet Instance initiates verification by preparing to check the Relying Party's identity via the Ident characteristic, that is a BLE GATT characteristic that carries an identifier value as described in [ISO18013-5 #11.1.3.2]. The Wallet Instance derives the expected Ident value and reads the Relying Party's Ident characteristic, comparing it to the expected Ident, and terminating the BLE connection if a mismatch may occur .

Note

The purpose of the Ident characteristic is only to verify whether the Wallet Instance is connected to the correct Relying Party Instance before starting data retrieval. If the Wallet Instance is connected to the wrong Relying Party Instance, session establishment will fail. Connecting and disconnecting to a Relying Party Instance takes a relatively large amount of time; therefore, it is faster to implement methods to identify the correct Relying Party Instance [ISO18013-5 #11.1.3.1] (*WP_112d* (page 754)).

Step 6: Relying Party Instance sends the encrypted `SessionEstablishment` message (includes `EReaderKey` and encrypted `DeviceRequest`) over the established BLE connection.

Step 7-8: [Optional] If the Wallet Unit receives the `SessionEstablishment` message during Negotiated Handover, the Wallet Unit MUST verify if this `SessionEstablishment` message matches the `SessionEstablishment` message received during the data retrieval phase (i.e., Step 6). In the event of a mismatch, the Wallet Unit MUST terminate the BLE connection [ISO18013-5 #9.2.3].

`SessionData` over BLE

The following figure illustrates the low-level flow compliant with ISO18013-5 for `SessionData` over BLE corresponding to Boxes C in Figure 8.10.

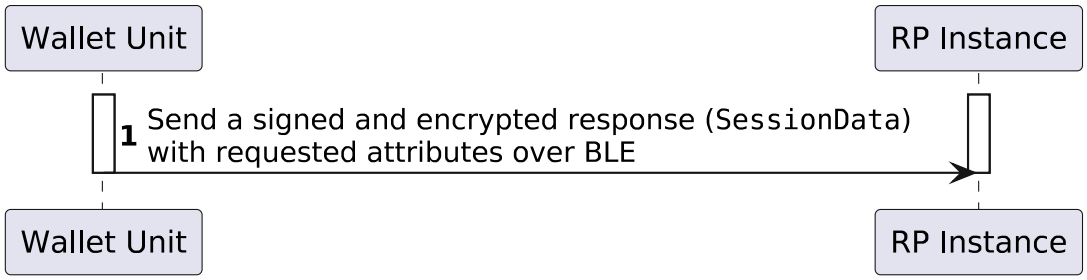


Fig. 12.19: Session Data over BLE.

Step 1: The Wallet Instance sends the final APDU containing either the last block Device Response (with requested attributes) or a status code, after which the session may end or continue with a new request.

`SessionEstablishment` over NFC

Note

If Device Engagement is initiated via a QR code, the Relying Party Instance has no standardised way to signal its intent to use NFC for subsequent data transfer. This could lead to a poor user experience, as the User might not be aware that they need to use NFC. This issue is avoided when NFC is used for the Device Engagement, as it implicitly establishes the data transfer method [ISO18013-5 #8.2.5].

Note

Due to the limited data transfer rate of NFC, if a large amount of data is required for a transaction, it may be neither practical nor reasonable for a User to keep the device within the RF range of the Relying Party Instance for the duration of the transaction. Furthermore, loss of signal when a device leaves the RF field necessitates re-initiating the transaction. This can only be avoided if all necessary User interactions with the Wallet Instance are performed while the device remains in the field, or if no User interaction is required during the active transmission phase [ISO18013-5 #8.2.5].

The following figure illustrates the low-level flow compliant with ISO18013-5 for `SessionEstablishment` over NFC corresponding to Box B in Figure 8.10.

Definitions (Acronyms and Commands)

Acronym/ Command	Description
PICC	Proximity Integrated Circuit Card, implemented by Wallet Units acting as a NFC tag.
PCD	Proximity Coupling Device, impletented by Relying Party instances using *APDU* exchanges.
AID	Application Identifier, unique ID used in SELECT *APDU* command to engage the Wallet Instance.
APDU	Application Protocol Data Unit, a standard message format for communication between Relying Party Instance (PCD) and Wallet (PICC) consists of Command APDUs (e.g., *SELECT*, *ENVELOPE*, *GET RESPONSE*) from the reader and Response *APDUs* from the wallet, which include data exchange and end exchange using status words (*SW1/SW2*).
SELECT APDU	Command that opens the Wallet Instance by *AID*. Response includes File Control Information (FCI) and Status Word (*SW1/SW2*).

continues on next page

Table 12.31 – continued from previous page

Acronym/ Command	Description
ENVELOPE APDU	Command that carries session messages (e.g., `SessionEstablishment`). Response indicates processing status (*OK* or *more data*).
GET RESPONSE APDU	Command that retrieves additional response data when the Wallet signals "more data" is available (*SW1=61*).
SW1/SW2	*SW1/SW2* (Status Words) — Two-byte status code at the end of every response. Common values: *90 00 = success*, *61 XX = more data*, *6A 82 = file/application not found*.

Step 1: The Relying Party Instance (PCD) sends a SELECT *APDU* command with the Application Identifier (*AID: A0 00 00 02 48 04 00*) to engage the Wallet Instance.

Step 2: The Wallet Instance (PICC) responds with File Control Information (FCI) and status words (SW1/SW2), either confirming success (*90 00*) or indicating that more data (*61 XX*) (*WP_112e* (page 754)).

Step 3: The Relying Party Instance sends an ENVELOPE *APDU* carrying the `SessionEstablishment` message, which contains the encrypted `DeviceRequest` and its ephemeral public key for session setup.

Step 4: The Wallet Instance processes `SessionEstablishment` and returns an *APDU* response with *SW1/SW2* (*OK* or *more data* to fetch), confirming the start of secure session context (*WP_112f* (page 754)).

Step 5-6: [Optional] The Wallet Instance receives the `SessionEstablishment` message during Negotiated Handover, the Wallet Instance MUST verify that this `SessionEstablishment` message matches the same message received during the data retrieval phase (i.e., Step 3-4). In the event of a mismatch, the Wallet Instance MUST terminate the NFC connection [ISO18013-5 #9.2.3].

`SessionData` over NFC

The following figure illustrates the low-level flow compliant with ISO18013-5 for `SessionData` over NFC corresponding to Box C in Figure 8.10.

Step 1-2: As long as the Wallet Instance signals that more data is available (*61 XX*), the Relying Party Instance issues *GET RESPONSE APDUs* to request the next block. The Wallet Instance returns encrypted `SessionData` fragments until all data is delivered.

Step 3: The Wallet Instance sends the final *APDU* containing either the last block Device Response (with requested attributes) or a status code, after which the session may end or continue with a new request.

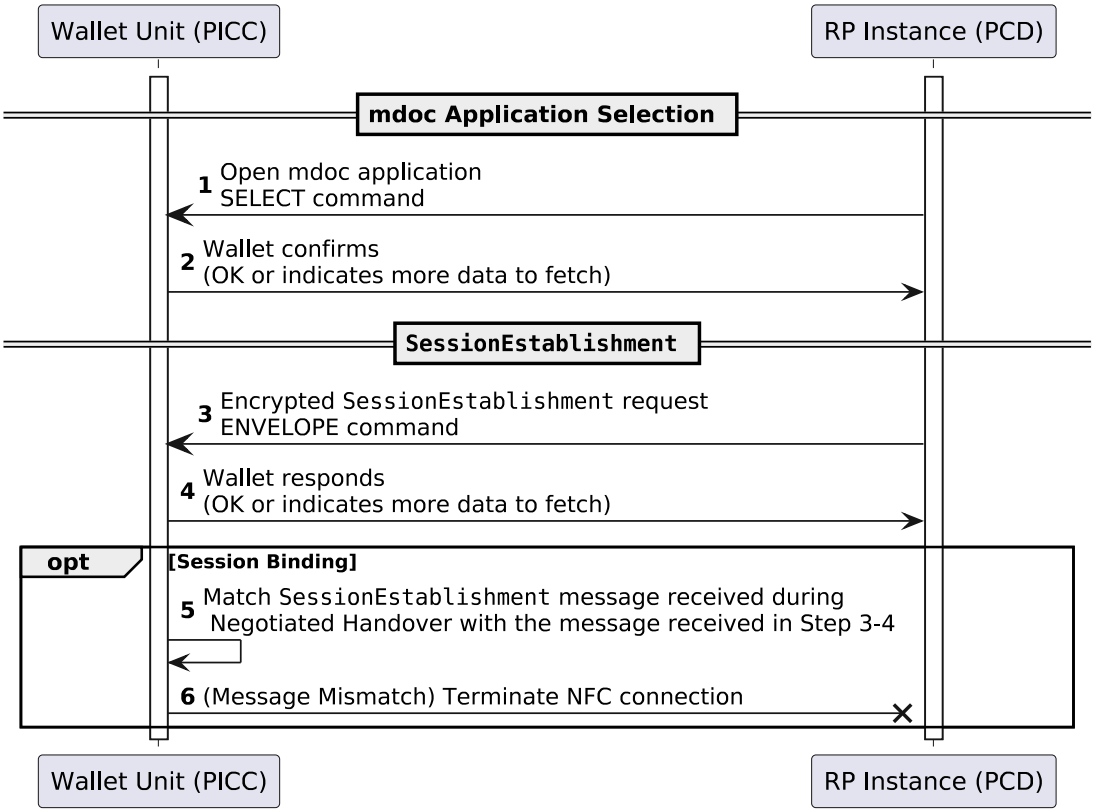


Fig. 12.20: Session Establishment over NFC.

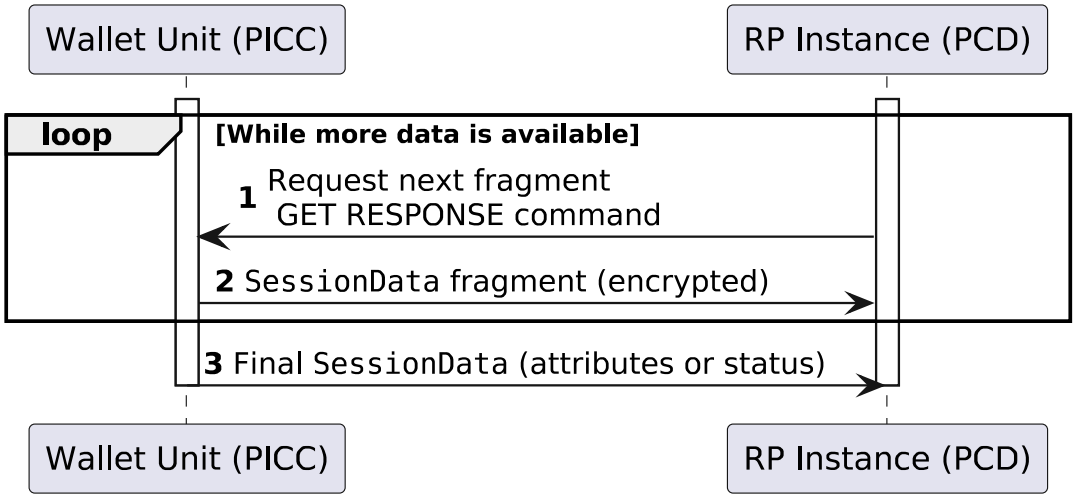


Fig. 12.21: Session Data over NFC.

Device Engagement

The Device Engagement structure MUST be CBOR encoded and have at least the following components (PPR-001, PPR-021, PPR-022, and *WP_102* (page 754)):

Component	Description
Version	*(tstr)*. Version of the Device Engagement structure.
Security	*(array)*. Contains two mandatory values: • *(int)*. Cipher suite identifier. See Table 22 of ISO18013-5. • *(bstr)*. Public ephemeral key generated by the Wallet Instance, used by the Relying Party Instance to derive the Session Key. The key MUST be of a type allowed by the selected cipher suite (PPR-022).
DeviceRetrievalMode-BLEOptions	*(map)*. Provides options for the BLE connection, such as Peripheral Server or Central Client mode, and the device UUID. See Table 2 of ISO18013-5 for the detailed mapping. If the Wallet Instance indicates during Device Engagement that it supports both modes, the Relying Party Instance SHOULD select the mdoc central client mode [ISO18013-5 #11.1.3.1]. Only present when performing Device Engagement using the QR code. Absent when using NFC to perform Device Engagement.
DeviceRetrievalMode-NFCOptions	*(map)*. Provides options for NFC connections, including the supported role (PICC or PCD) and maximum PDU command/response sizes. See Table 2 of ISO18013-5 for the detailed mapping. In case NFC is used for Device Retrieval, the Wallet Instance MUST support PICC mode and the Relying Party Instance MUST support PCD mode [ISO18013-5 #11.2]. Only present when performing Device Engagement using the QR code. Absent when using NFC to perform Device Engagement.
Capabilities	*(map)*. Declares optional capabilities supported by the mdoc, that are: • **HandoverSessionEstablishmentSupport** *(bool)*. If present, it MUST be set to *true*. Indicates support for receiving the *SessionEstablishment* message during Negotiated Handover, as defined in [ISO18013-5 #9.2.3] (PPR-024). • **ReaderAuthAllSupport** *(bool)*. If present, it MUST be set to *true*. Indicates support for receiving the *ReaderAuthAll* structure in the mdoc request, as defined in [ISO18013-5 #10.2.6] (PPR-025).

continues on next page

Table 12.32 – continued from previous page

Component	Description
OriginInfos	*(array)*. Describes the interface used to receive and deliver the engagement structure. *OriginInfos* MAY be an empty array.

mdoc Request

The messages in the mdoc Request MUST be encoded using CBOR. The resulting CBOR byte string for the mdoc Request MUST be encrypted with the Session Key obtained after the Device Engagement phase and MUST be transmitted using the BLE or NFC protocol (PPR-026, PPR-027, PPR-028).

Each mdoc Request MUST be compliant with the following structure, and MUST include the following components, unless otherwise specified:

Component	Description
version	*(tstr)*. Version of the mdoc Request structure. Enables compatibility management across different versions or implementation profiles.
docRequests	*(array)*. Each entry is a *DocRequest* containing: • **itemsRequest**. CBOR-encoded *ItemsRequest* structure, formatted as: – **docType** *(tstr)*. The type of document requested. See *mdoc-CBOR Credential Format* (page 366). – **nameSpaces** *(map)*. A map of namespace identifiers to requested *DataElements*. Each entry in *DataElements* includes: ∗ **DataElementIdentifier** *(tstr)*. The identifier of the requested data element. ∗ **IntentToRetain** *(bool)*. Indicates whether the Relying Party intends to retain the value of the data element. • **readerAuth** *(COSE_Sign1, CONDITIONAL)*. Used to authenticate the Relying Party Instance for each *DocRequest*. The signature is computed over *ReaderAuthentication* data, as defined in [ISO18013-5 #12.5]. This component MUST be present only if *readerAuthAll* is not used (PPR-025).

continues on next page

Table 12.33 – continued from previous page

Component	Description
readerAuthAll	*(COSE_Sign1, CONDITIONAL).* Used to authenticate the Relying Party once for all *DocRequest's. The signature is computed over 'ReaderAuthenticationAll* data, as defined in [ISO18013-5 #12.5]. This component MUST be present only if *ReaderAuthAllSupport* is set to *true* in the DeviceEngagement structure, and individual *readerAuth* fields are not used (PPR-025).

mdoc Response

The messages in the mdoc Response MUST be encoded using CBOR and MUST be encrypted with the Session Key obtained after the Device Engagement phase (PPR-029, PPR-030).

Each mdoc Response MUST be compliant with the following structure, and MUST include the following components, unless otherwise specified:

Component	Description
version	*(tstr).* Version of the mdoc Response structure. Enables tracking changes and maintaining compatibility across versions of the standard or implementation profiles.
documents	*(array of Documents, OPTIONAL).* CBOR-encoded collection of documents returned in response to the request. Each document includes *issuerSigned* and *deviceSigned* components, and follows the structure defined in [ISO18013-5 # 10.3.3].
documentErrors	*(map, OPTIONAL).* A map of error codes for unreturned documents, as defined in [ISO18013-5 #10.3.6]. Each key is a *docType*, and each value is an *ErrorCode* (int) indicating the reason why the document was not returned.
status	*(uint).* Status code indicating the outcome of the request. For example, *"status": 0* means successful processing. For details, see Table 3 (ResponseStatus) of [ISO18013-5 #10.3.5].

Each document in **documents** MUST be compliant with the following structure, and it MUST include the following components, unless otherwise specified (PPR-029):

Component	Description
docType	*(tstr).* Document type identifier. For example, for an mDL, the value MUST be `org.iso.18013.5.1.mDL`.

continues on next page

Table 12.35 – continued from previous page

Component	Description
issuerSigned	*(bstr).* Contains the *IssuerNameSpaces* structure, which includes data elements signed by the Issuer, and the *issuerAuth* structure, which ensures their authenticity and integrity using the Mobile Security Object (MSO). See *mdoc-CBOR Credential Format* (page 366).
deviceSigned	*(bstr).* Contains the *DeviceNameSpaces* structure (data elements signed by the Wallet Instance), and the *deviceAuth* structure, which includes the authentication data signed by the Wallet Instance. See the table below for details.
errors	*(map, OPTIONAL).* A map of error codes for each unreturned data element grouped by namespace. Each key represents a namespace, and each value is a map of data element identifiers to corresponding error codes. See [ISO18013-5 #10.3.6] for details on the errors structure.

A **deviceSigned** data structure MUST be compliant with the following structure (*WP_111a* (page 754)), and MUST include the following components:

Component	Description
nameSpaces	*(bstr).* Contains the *DeviceNameSpaces* structure. It MAY be an empty structure. *DeviceNameSpaces* maps namespace identifiers to a set of data elements signed by the Wallet Instance. Each namespace contains one or more *DeviceSignedItem*, where each item includes: • **DataItemName** *(tstr).* The identifier of the data element. • **DataItemValue** *(any).* The value of the data element.
deviceAuth	*(COSE_Sign1).* Contains the *DeviceAuth* structure, which MUST include the **deviceSignature** for the Wallet Instance authentication. The signature is computed over the *DeviceAuthentication* data, which binds the returned elements to the session and the request. See [ISO18013-5 #12.4] for details on the authentication structure.

Session Termination

The session MUST be terminated if at least one of the following conditions occur (PPR-007 and *WP_113–113a* (page 754)):

- After a time-out of no activity of receiving or sending session establishment or session data messages occurs. The time-out for no activity implemented by the Wallet Instance and the Relying Party Instance SHOULD be no less than 300 seconds;

- When the Wallet Instance does not accept any more requests;
- When the Relying Party Instance does not send any further requests.

If the Wallet Instance and the Relying Party Instance do not send or receive any further requests, the session termination MUST be initiated as follows (PPR-007 and *WP_113* (page 754)):

- Send the status code for session termination, or
- Dispatch the "End" command as outlined in [ISO18013-5 #11.1.3.3].

When a session is terminated, the Wallet Instance and the Relying Party Instance MUST perform at least the following actions (*WP_114* (page 754)):

- Destruction of session keys and related ephemeral key material;
- Closure of the communication channel used for data retrieval.

Note

See *mdoc-CBOR Credential Format* (page 366) for the meaning of CBOR type acronyms.

12.2.3 Identity Matching and Identity Reconciliation

Identity matching is the process by which a Relying Party establishes that the person identification attributes presented in a transaction refer to the same natural person. It is the same practical approach used in any legacy authentication and authorization infrastructure. It is not a Wallet Instance feature, nor a step of the presentation protocol. Cryptographic verification of the Digital Credential remains a distinct prerequisite, and identity reconciliation is a subsequent, optional step.

After cryptographic verification of a presented Digital Credential, in remote or proximity flows, the Relying Party MUST first perform **identity matching**, to establish that the person identification attributes presented in the current transaction refer to the same natural person. Only after a successful identity matching, the Relying Party MAY perform **identity reconciliation**, linking that natural person to a previous User session or stored User record.

If identity matching fails, the Relying Party MUST NOT perform identity reconciliation and MUST NOT treat the presenter as a previously known User.

This Section specifies the matching and reconciliation rules that apply to person identification Digital Credentials (PID and IT-Wallet ID). Correlation of identities during IT-Wallet ID issuance (IdP–MRTD) is specified in *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457) and is out of scope of this Section.

Identity Matching

Identity matching determines whether the presented person identification attributes identify one natural person.

The Relying Party MUST apply the matching pattern of the presented Digital Credential type, as defined below. Different Digital Credential types MAY define different identity matching patterns. For the PID User attributes, see *PID Data Model* (page 380). For the IT-Wallet ID User attributes, see *IT-Wallet ID Data Model* (page 392).

Identity Reconciliation

Identity reconciliation is the subsequent step that binds a successfully matched natural person to a Relying Party User record or session.

The Relying Party MUST NOT complete identity reconciliation unless identity matching has succeeded. Reconciliation MAY create a new User record when no stored record matches.

Matching Patterns by Digital Credential Type

Unique National Identifier Binding

When `personal_administrative_number` or `tax_id_code` is presented, the Relying Party MUST use it as the primary identity matching key before any identity reconciliation with a stored User record (*CI_201* (page 776)).

For the IT-Wallet ID, at least one of `tax_id_code` and `personal_administrative_number` is present. The Relying Party MUST use the unique national identifier that is presented as the primary matching key.

Attribute-Based Binding

For the PID, `personal_administrative_number` is OPTIONAL. Relying Parties MUST NOT assume that a unique national identifier is always available.

When `personal_administrative_number` is not presented, the Relying Party MUST perform *Attribute-Based Binding* by matching at least `family_name`, `given_name`, `birth_date` (or `birth-date`) and `place_of_birth` after normalisation of case, whitespace and diacritics. The Relying Party MUST apply the same normalisation to the presented attributes and to the stored User record, including the string members of `place_of_birth` (`country`, `region`, `locality`). Comparison of the date of birth MUST use the ISO 8601 calendar date (`YYYY-MM-DD`), independent of display format.

The Relying Party MUST NOT complete identity matching on a subset of those attributes that would allow a person swap (*CI_202* (page 776)). In particular:

- matching only a pair of names, only a name and a date of birth, or any subset that omits `place_of_birth`, MUST NOT be treated as a successful identity match;
- a mismatch on any attribute of the minimum set MUST fail identity matching; the Relying Party MUST NOT ignore a current-name mismatch (for example after a name change) in order to force a match;
- if Attribute-Based Binding matches more than one stored User record, the Relying Party MUST NOT complete identity matching. It MAY request a unique national identifier, or apply a supplementary binding method that uniquely identifies the person.

Supplementary Binding Methods

The following methods MAY supplement Attribute-Based Binding. They MUST NOT replace the Attribute-Based Binding minimum set when no unique national identifier is presented:

- **Session-Based Binding**: the presentation occurs within an authenticated session already bound to the User;
- **Issuer-Attested Binding**: the Issuer attests that the presented attributes belong to the same subject (for example through a stable Issuer subject identifier);
- **Relying Party-Specific Identifiers**: identifiers previously issued by the Relying Party to that User after a successful identity match;
- **Cryptographic Binding**: proving that the private keys of the presented Digital Credentials are managed by the same Keystore or WSCD.

Privacy and Selective Disclosure

Selective disclosure MUST be used so that identifying attributes needed only for matching are not released when not required for the specific transaction, in line with Article 5a of the European Digital Identity Regulation and the privacy-preserving combined presentation principles of the ARF, to be fully applicable when the IT-Wallet will be notified as an EUDIW solution.

Application to User Attribute Deletion

When the User requests deletion of presented attributes, the Relying Party MUST uniquely identify one or more Digital Credentials by applying identity matching, as specified in *User's Attributes Deletion* (page 297).

When the User authenticates at the Erasure Endpoint by presenting a Digital Credential from the Wallet Instance, the Relying Party MUST apply the matching patterns of this Section.

When the User authenticates at the Erasure Endpoint without using the Wallet Instance, the Relying Party MAY perform identity matching and identity reconciliation using person identification attributes obtained from a preexisting national authentication scheme (for example Cie ID), where those attributes uniquely identify the User in the Relying Party's records.

12.3 Wallet Metadata Retrieval Flow

There are Digital Credential flows that require retrieving Wallet information before interacting with the Wallet itself. For example, to enable the Wallet Solution selection option during a Credential Offer flow or a Presentation flow. To do so, the third parties offering the Credential Offer (e.g., Authentic Source, Registry, Catalogue) and the Relying Parties consult the IT-Wallet System Register, in particular the Federation Registry, via the Trust Anchor's endpoints.

First, using the Immediate Subordinate Listing endpoint they obtain the list of all subordinates of the Trust Anchor identified by their `entity_id`. To obtain only the Wallet Provider `entity_id`, a query parameter `entity_type` set with `openid_wallet_provider` MAY be used. If the responder does not support this feature, it MUST use the HTTP status code 400 and the content type `application/json`, with the error code `unsupported_parameter`.

Then, given the `entity_id`, they fetch the corresponding Entity Configuration for each Wallet Provider using the federation metadata endpoint (GET .well-known/openid-federation) and obtain the Wallet Solution metadata (see *Wallet Solution Metadata* (page 308)). More details on the use of the Trust Anchor's endpoints are provided in *The Infrastructure of Trust* (page 109).

Inside the metadata, the third parties offering the Credential Offer and the Relying Parties retrieve the necessary information for the Selection Page, i.e. the logo and name in full of the Wallet Solutions. More details on the design of the Selection Page are provided in *User Experience Design* (page 35).

CHAPTER

THIRTEEN

ENDPOINTS

13.1 Wallet Provider Endpoints

The Wallet Provider, responsible for delivering a Wallet Solution, MUST expose the endpoints to support trust establishment and essential Wallet Instance functionalities. These include the `/.well-known/openid-federation` Federation Endpoint which MUST adhere to the OpenID Federation 1.0 specification to reliably establish trust with the Wallet Provider's as well as, endpoints for Wallet Instance registration, nonce generation (required for registration), attestation issuance, and revocation. Aside from the Federation endpoint, the implementation details of the others are left to the Wallet Provider's discretion.

Note

Tests related to the use of Wallet Provider endpoints are defined in *Wallet Provider Test Matrix* (page 736), particularly *Test Cases for Wallet Provider Backend* (page 737), *Test Cases for Wallet Instance* (page 739), and *Optional Test Cases for Wallet Instance* (page 767).

13.1.1 Federation Endpoint

The `/.well-known/openid-federation` endpoint serves as the discovery mechanism for trust establishment by retrieving the Wallet Provider Entity Configuration.

See Section *Wallet Provider Entity Configuration* (page 305) for technical details (*WP_001–004* (page 737)).

13.1.2 Wallet Solution Nonce Endpoint

This is a RESTful API endpoint that allows the Wallet Instance to request a cryptographic nonce from the Wallet Provider. The nonce serves as an unpredictable, single-use challenge to ensure freshness and prevent replay attacks.

See *Mobile Application Nonce Request* (page 672) and *Mobile Application Nonce Response* (page 672) for details on the Nonce Request and Nonce Response (*WP_131* (page 767)).

13.1.3 Wallet Instance Management Endpoint

These are RESTful API endpoints provided by the Wallet Provider that enables Wallet Instance management, including registration, status retrieval, revocation upon request (e.g., by the User), and deletion. The following sections describe the registration, status retrieval and revocation requests, along with their corresponding responses, handled by this endpoint, which are required for core *Wallet Instance Functionalities* (page 284).

Wallet Instance Registration Request

To register a Wallet Instance, the request to the Wallet Provider MUST use the HTTP POST method with `Content-Type` set to *application/json*. The request body MUST contain the claims described in *Mobile Application Instance Initialization Request* (page 673) (*WP_131–134* (page 767)).

Wallet Instance Registration Response

If a Wallet Instance Registration Request is successfully validated, the Wallet Provider provides an HTTP Response with status code 204 (No Content). For detatails see *Mobile Application Instance Initialization Response* (page 674) (*WP_135–137* (page 767)).

Wallet Instance Retrieval Request

To retrieve all Wallet Instances associated with a User, a request MUST be sent using the HTTP GET method to the Wallet Provider (*WP_145* (page 767)).

Note

For retrieving a specific Wallet Instance, the request MUST include the Wallet Instance ID as a path parameter.

Wallet Instance Retrieval Response

If a Wallet Instance Retrieval Request is successfully processed, the Wallet Provider MUST return an HTTP Response with a 200 (OK) status code. The response body MUST be in JSON format and include the relevant Wallet Instance information, such as its unique ID, status, and issuance date. When retrieving all Wallet Instances, the response MUST return an array containing the details of all associated instances (*WP_146* (page 767)).

If any errors occur during the retrieval process, an error response MUST be returned. Refer to *Error Handling for Wallet Instance Management* (page 546) for details on error codes and descriptions.

Below is a non-normative example of an error response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store

{
  "error": "forbidden",
  "error_description": "User is not authorized to retrieve Wallet Instances."
}
```

Wallet Instance Revocation Request

To revoke an active Wallet Instance, a revocation request MUST be sent using the HTTP PATCH method with Content-Type set to `application/json`. The request body MUST contain a `status` parameter set to `REVOKED` (*WP_147* (page 767)).

Note

While PATCH is the recommended method, the revocation request MAY also be sent using the POST method, depending on implementation preferences.

Wallet Instance Revocation Response

If a Wallet Instance Revocation Request is successfully processed, the Wallet Provider provides an HTTP Response with a 204 (No Content) status code (*WP_148* (page 767)).

If any errors occur during the Wallet Instance Revocation, an error response MUST be returned. Refer to *Error Handling for Wallet Instance Management* (page 546) for details on error codes and descriptions (*WP_035–039, WP_043–044* (page 739)).

Below is a non-normative example of an error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "bad_request",
  "error_description": "The request is missing status parameter."
}
```

Error Handling for Wallet Instance Management

To ensure robustness and security, the Wallet Provider MUST handle errors consistently across all Wallet Instance Management requests, including Registration, Retrieval, and Revocation.

In case of an error, the Wallet Provider MUST return an error response as defined in **RFC 7231**, with additional details available in **RFC 7807**. The response MUST use the Content-Type set to `application/json` and MUST include the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

The following sections categorize errors into **common errors**, which apply to all requests, and **request-specific errors**, which are relevant to particular operations (*WP_035–044* (page 739), and *WP_150–155* (page 767)).

Common Error Responses

The following errors apply to all Wallet Instance Management operations (Registration, Retrieval, and Revocation), and MUST be supported for the error response, unless otherwise specified (*WP_035–039* (page 739)):

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters, or includes invalid and unknown parameters.
`422 Unprocessable Content` [OPTIONAL]	`validation_error`	The request does not adhere to the required format.
`500 Internal Server Error`	`server_error`	An internal error occurred while processing the request.
`503 Service Unavailable`	`temporarily_unavailable`	The service is unavailable. Please try again later.

Request-Specific Error Responses

The errors in *Mobile Application Instance Initialization Error Response* (page 674) MUST be supported for error responses related to **Wallet Instance Registration**.

The following errors MUST be supported for error responses related to **Wallet Instance Retrieval** (*WP_041–042* (page 739)):

HTTP Status Code	Error Code	Description
`403 Forbidden`	`forbidden`	The user does not have permission to retrieve this Wallet Instance.
`401 Unauthorized`	`unauthorized`	The request lacks valid authentication credentials.

The following errors MUST be supported for error responses related to **Wallet Instance Revocation** (*WP_043–044* (page 739)):

HTTP Status Code	Error Code	Description
`403 Forbidden`	`invalid_request`	The user does not have permission to revoke this Wallet Instance.
`401 Unauthorized`	`unauthorized`	The request cannot be authenticated or authorized.

13.1.4 Wallet Instance Attestation Issuance Endpoint

This is a RESTful API endpoint provided by the Wallet Provider that enables the Wallet Instance to obtain Wallet Instance Attestation, by sending a Wallet Instance Attestation Issuance Request.

Wallet Instance Attestation Issuance Request

The Wallet Instance Attestation Issuance Request uses the HTTP POST method with `Content-Type` set to `application/json`. (*WP_026* (page 739) and *WP_140–142* (page 767)).

The `typ` header of the Wallet Instance Attestation Issuance Request JWT assumes the value `wia-request+jwt`.

The Wallet Instance Attestation Issuance Request body contains an `assertion` parameter whose value is a signed JWT including all header parameters and body claims described below.

Below is a non-normative example of a Wallet Instance Attestation Request.

```
POST /wallet-instance-attestation HTTP/1.1
Host: application-provider.example.org
Content-Type: application/json
```

```
{
  "assertion": "eyJpc3MiOiJPbnNpYW5kcklqcDdJbU55ZGlJNklsQ..."
}
```

In particular, the Wallet Instance Attestation Issuance JWT includes the following HTTP header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	[**RFC 7516#section-4.1.1**]
kid	Thumbprint of the Wallet Instance's JWK contained in the `cnf` claim.	[**RFC 7638#section_3**]
typ	The type of the JWT, it MUST set to `wia-request+jwt`.	

The Wallet Instance Attestation Request JWT includes the following body claims:

Claim	Description	Reference
iss	String containing the unique identifier of the Wallet Instance.	[**RFC 9126**], [**RFC 7519**].
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 9126**], [**RFC 7519**].
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 9126**], [**RFC 7519**].
nonce	The `nonce` obtained from the Nonce Endpoint.	
hardware_signature	The signature of `client_data_hash` obtained using the Cryptographic Hardware Key, encoded in the `base64url` format.	
integrity_assertion	The Integrity Assertion for Wallet Instance Attestation obtained from the **Device Integrity Service APIs** with the holder binding of `client_data_hash`.	
hardware_key_tag	The value of the Cryptographic Hardware Key Tag.	
cnf	JSON object containing the public part of an asymmetric key pair owned by the Wallet Instance.	**RFC 7800**.
platform	String containing the value of the device operating system.	
wallet_solution_id	String containing the identifier of the Wallet Solution.	
wallet_solution_version	String containing the version of the Wallet Solution.	

Below is a non-normative example of a Wallet Instance Attestation Request JWT header and payload.


```
{
  "alg": "ES256",
  "kid": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "typ": "wia-request+jwt"
}
```

```
{
  "iss": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
  "hardware_signature": "KoZIhvcNAQcCoIAwgAIB...",
  "integrity_assertion": "o2NmbXRvYXBwbGUtYXBwYXNzZXJ0aW9uLXBheWxvYWQtYXBw...",
  "hardware_key_tag": "QW12DylRTmF89iGkpydNDWW7m8bVpa2Fn9KBeXGYtfX",
  "cnf": {
    "jwk": {
      "crv": "P-256",
      "kty": "EC",
      "x": "8FJtI-yr3pjyRKGMnz4WmdnQD_uJSq4R95Nj98b44",
      "y": "MKZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg"
    }
  },
  "platform": "iOS",
  "wallet_solution_id": "Wallet-mobile",
  "wallet_solution_version": "1.1.0"
}
```


Wallet Instance Attestation Issuance Response

If the Wallet Instance Attestation Issuance Request is successfully validated, the Wallet Provider returns an HTTP response with a status code of `200 OK` and `Content-Type application/json`. The returned JSON Object includes `wallet_instance_attestation` (see *Wallet Instance Attestation Issuance* (page 285)). `wallet_instance_attestation` is signed by the Wallet Provider (*WP_027–029* (page 739) and *WP_143–144* (page 767)). The JWT formatted Wallet Instance Attestation is to be used for the Issuance phase, as an OAuth Client Attestation, and will be sent to the Credential Issuer as discussed in *Digital Credential Issuance* (page 419).

The JSON Object returned in the response has the following claim:

Parameter	Description	Reference
wallet_instance_attestation	REQUIRED. A String containing of the issued Wallet Instance Attestation.	This specification.

The value of `wallet_instance_attestation` parameter is a string representing the Wallet Instance Attestation in a JWT.

If any errors occur during the process, an error response is returned. The response uses `application/json` as the `Content-Type` and includes the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered (*WP_035* (page 739)).

Below is a non-normative example of a Wallet Instance Attestation Issuance Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The provided challenge is invalid, expired, or already used."
}
```

The following table lists HTTP Status Codes and related error codes that are supported for the error response, unless otherwise specified (*WP_036–039* (page 739) and *WP_150–155* (page 767)):

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters (e.g., header parameters, or Integrity Assertion), or includes invalid and unknown parameters.
`403 Forbidden`	`invalid_request`	The Wallet Instance has been revoked.
`403 Forbidden`	`integrity_check_error`	The device does not meet the Wallet Provider's minimum security requirements.
`403 Forbidden`	`invalid_request`	The signature of the Wallet Instance Attestation Request is invalid or does not match the associated public key (JWK).
`403 Forbidden`	`invalid_request`	The Integrity Assertion validation failed; the Integrity Assertion is tampered with or improperly signed.
`403 Forbidden`	`invalid_request`	The provided `nonce` is invalid, expired, or already used.

continues on next page

Table 13.7 – continued from previous page

HTTP Status Code	Error Code	Description
`403 Forbidden`	`invalid_request`	The Proof of Possession (`hardware_signature`) is invalid.
`403 Forbidden`	`invalid_request`	The `iss` parameter does not match the Wallet Provider's expected URL identifier.
`404 Not Found`	`not_found`	The Wallet Instance was not found.
`422 Unprocessable Content` [OPTIONAL]	`validation_error`	The request does not adhere to the required format.
`500 Internal Server Error`	`server_error`	An internal server error occurred while processing the request.
`503 Service Unavailable`	`temporarily_unavailable`	The service is unavailable. Please try again later.

Wallet Instance Attestation JWT

The JOSE header of the Wallet Instance Attestation JWT contains the following parameters:

JOSE header	Description	Reference
alg	REQUIRED. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	**RFC 7516#section-4.1.1**.
kid	REQUIRED. Unique identifier of the public key associated to the private key the Wallet Provider used to sign the Wallet Instance Attestation.	**RFC 7638#section_3**.
typ	REQUIRED. It MUST be set to `oauth-client-attestation+jwt`	OPENID4VC-HAIP.
trust_chain	OPTIONAL. Sequence of Entity Statements that composes the Trust Chain related to the Wallet Provider.	OID-FED Section 4.3 *Trust Chain Header Parameter*.
x5c	REQUIRED. Contains the X.509 public key certificate or certificate chain (**RFC 5280**) corresponding to the key used to digitally sign the JWT.	**RFC 7515** Section 4.1.8, SD-JWT-VC Section 3.5 and OPENID4VC-HAIP.

The body of the Wallet Instance Attestation JWT contains the following claims:

Claim	Description	Reference
iss	REQUIRED. String containing the URL identifying the Wallet Provider.	**RFC 7519**.
sub	REQUIRED. JWK Thumbprint of the public key included in the `cnf` claim.	**RFC 7519** and EUDI-TS 3.
iat	REQUIRED. UNIX Timestamp with the time of JWT issuance.	**RFC 9126** and **RFC 7519**.
exp	REQUIRED. UNIX Timestamp with the expiry time of the JWT. The lifetime is specified in *the Wallet Instance Attestation lifetime* (page 272).	**RFC 9126** and **RFC 7519** and EUDI-TS 3.
nbf	OPTIONAL. UNIX Timestamp with the start time of validity of the JWT issuance.	**RFC 9126** and **RFC 7519**.
cnf	REQUIRED. JSON object, containing the public part of an asymmetric key pair owned by the Wallet Instance.	**RFC 7800**.
wallet_link	REQUIRED. String containing a URL to get further information about the Wallet and the Wallet Provider.	OpenID4VCI.
wallet_name	REQUIRED. String containing a human-readable name of the Wallet.	OpenID4VCI.

Below is a non-normative example of the Wallet Instance Attestation JWT header and payload, without encoding and signature applied:


```
{
    "typ": "oauth-client-attestation+jwt",
    "x5c": [
      "MIICrDCCAlKgAwIBAgIUCQ5zD8eGxAN6c5isgQk0jkp3qzwwCgYIKoZIzj0EAwIwgZIxCzAJBgNVBAYTAklUMQ4w
DAYDVQQIDAVMYXppbzENMAsGA1UEBwwEUm9tYTEWMBQGA1UECgwNUGFnb1BBIFMucC5BLjEkMCIGA1UEAwwbZm9vMTEuYmx
vYi5jb3JlLndpbmRvd3MubmV0MSYwJAYJKoZIhvcNAQkBFhdwYWdvcGFzcGFAcGVjLnBhZ29wYS5pdDAeFw0yNTA3MDMxNT
E2NTlaFw0yNjA3MDMxNTE2NTlaMIGSMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNVBAcMBFJvbWExFjAUB
gNVBAoMDVBhZ29QQSBTLnAuQS4xJDAiBgNVBAMMG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDEmMCQGCSqGSIb3DQEJ
ARYXcGFnb3Bhc3BhQHBlYy5wYWdvcGEuaXQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARtdnuBd5hAg3cPyz7/o8vBFyL
4sp45HikFogMZse7f9/iL4gn6YM/ehP1CQI0dFnT3c/RdxFyRukKWXscopwXso4GDMIGAMAkGA1UdEwQCMAAwCwYDVR0PBA
QDAgWgMCYGA1UdEQQfMB2CG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDAdBgNVHQ4EFgQUOd7M/0bo1+saPT/H9M+G9
6sKnYIwHwYDVR0jBBgwFoAUkKHxQLaudIm0WfwZeN5HMTG8SCMwCgYIKoZIzj0EAwIDSAAwRQIgSXKs9Gx2bXVG9nxap1I/
KqwHdh6SQDKI85J9n7FqrI8CIQCqltHTw6sci1R4RSOFySEvzEohQTbNTke+X2lD9iQd5A==",
```

```
      "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0OTMzWhcNMjcwNjA0MTI0OTMzWjCBkjELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMRYwFAYDVQQKDA1QYWdvUEEgUy5wLkEuMSQwIgYDVQQDDBtmb28xMS5ibG9iLmNvcmUud2luZG93
cy5uZXQxJjAkBgkqhkiG9w0BCQEWF3BhZ29wYXNwYUBwZWMucGFnb3BhLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA
EgXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l/iX7p1ysisVV2vOhj1xEm/3M24U5HcywPPpNIOEHTESjqOCAW4wgg
FqMB0GA1UdDgQWBBSQofFAtq50ibRZ/Bl43kcxMbxIIzCB5QYDVR0jBIHdMIHagBTzUINxjECpXem0zyIpsnTqv3DEGaGBu
aSBtjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQ
b2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYDVQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHp
zLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5pcHpzLml0ggYBlzr5eUMwEgYDVR0TAQH/BAgwBgEB/wIBAD
AOBgNVHQ8BAf8EBAMCAAcwPQYDVR0fBDYwNDAyoDCgLoYsaHR0cHM6Ly9wcmUudGEud2FsbGV0LmlwenMuaXQvcGtpL3RhL
XN1Yi5jcmwwCgYIKoZIzj0EAwIDRwAwRAIgBSyH++oPDP8ipRxHcJ8u2HkN7wL44XpJFxSIBV9RFHMCIBixpcT43NSsk0Ev
hJb4g68c1d9OrBnXOXVuyCtWWE6a",
      "MIIDQzCCAuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0NTE3WhcNMzAwNjAzMTI0NTE3WjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQb2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYD
VQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHpzLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5
pcHpzLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEaE0xyhd3e9LDT7uwHOclL5H3389gwiCwFhI3KOvidn0glBIHYx
qH+4Z9VTMYWG5L8cwC9AaJUCNGu2dp5ZiiTKOB5TCB4jAdBgNVHQ4EFgQU81CDcYxAqV3ptM8iKbJ06r9wxBkwHwYDVR0jB
BgwFoAU81CDcYxAqV3ptM8iKbJ06r9wxBkwDwYDVR0TAQH/BAUwAwEB/zBEBggrBgEFBQcBAQQ4MDYwNAYIKwYBBQUHMAKG
KGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jZXIwDgYDVR0PAQH/BAQDAgEGMDkGA1UdHwQyMDAwLqA
soCqGKGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jcmwwCgYIKoZIzj0EAwIDSQAwRgIhAOsQYzR+eG
f4je63VGHqkpmkBbfyOre+mfIdHHowWWR/AiEA58xBNb5UW5uMB+tQur8fq24RD5MmRHLYS6bDgIYmluw="
    ],
    "alg": "ES256",
    "kid": "8UkfrvttLkpAQOOp"
  }

{
    "cnf": {
      "jwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "8g9qKk3tRCBHsu57pNy3oMPc-jeJW8zU1LvfG9H5__M",
        "y": "KD0MNCPtJqUd1RGiTdAxEL6OCw5trfzqSrU6BHKOy2g",
        "kid": "dd762e6f8b613193",
        "alg": "ES256"
      }
    },
    "iss": "https://wallet-provider.example.org",
    "sub": "dd762e6f8b613193",
    "wallet_name": "ExampleOrg",
    "wallet_link": "https://example.org/wallets/ExampleOrg/info",
    "iat": 1772444318,
```

```
  "exp": 1772447918
}
```

Note

As a revocation mechanism for WIA, the per-issuer reuse option described in Section 2.5.1 of EUDI-TS 3 is preferred.

13.1.5 Key Attestation Issuance Endpoint

This is a RESTful API endpoint provided by the Wallet Provider that enables the Wallet Instance to obtain Key Attestation, by sending a Key Attestation Issuance Request.

Key Attestation Issuance Request

The Key Attestation Issuance Request uses the HTTP POST method with `Content-Type` set to `application/json`. (*WP_026* (page 739) and *WP_140–142* (page 767)).

The `typ` header of the Key Attestation Issuance Request JWT assumes the value `wua-request+jwt`.

The Key Attestation Issuance Request body contains an `assertion` parameter whose value is a signed JWT including all header parameters and body claims described below.

Below is a non-normative example of a Key Attestation Request.

```
POST /key-attestation HTTP/1.1
Host: application-provider.example.org
Content-Type: application/json

{
  "assertion": "eyJpc3MiOiJPbnNpYW5kcklqcDdJbU55ZGlJNklsQ..."
}
```

In particular, the Key Attestation Issuance JWT includes the following HTTP header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	[**RFC 7516# section-4.1.1**]

continues on next page

Table 13.10 – continued from previous page

Parameter	Description	Reference
kid	Thumbprint of the Wallet Instance's JWK contained in the `cnf` claim.	[RFC 7638#section_3]
typ	The type of the JWT, it MUST set to `wua-request+jwt`.	

The Key Attestation Request JWT includes the following body claims:

Claim	Description	Reference
iss	String containing the unique identifier of the Wallet Instance.	[RFC 9126], [RFC 7519].
exp	UNIX timestamp representing the JWT expiration time.	[RFC 9126], [RFC 7519].
iat	UNIX timestamp representing the JWT issuance time.	[RFC 9126], [RFC 7519].
nonce	The `nonce` obtained from the Nonce Endpoint.	
keys_to_attest	JSON array of JWT strings, each representing a `Key_Attestation_Requests`.	
hardware_signature	The signature of `client_data_hash` obtained using the Cryptographic Hardware Key, encoded in the `base64url` format.	
integrity_assertion	The Integrity Assertion for Wallet Instance Attestation obtained from the **Device Integrity Service APIs** with the holder binding of `client_data_hash`.	
hardware_key_tag	The value of the Cryptographic Hardware Key Tag.	
cnf	JSON object containing the public part of the first asymmetric key pair (first element of `keys_to_attest`) owned by the Wallet Instance.	RFC 7800.
platform	String containing the value of the device operating system.	
wallet_solution_id	String containing the identifier of the Wallet Solution.	
wallet_solution_version	String containing the version of the Wallet Solution.	

Below is a non-normative example of a Key Attestation Request JWT header and payload.

```
{
  "alg": "ES256",
  "kid": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "typ": "wua-request+jwt"
}
```

```
{
  "iss": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
```

```
  "hardware_signature": "KoZIhvcNAQcCoIAwgAIB...",
  "integrity_assertion": "o2NmbXRvYXBwbGUtYXBwYXNzZXJ0aW9uLXBheWxvYWQtYXBw...",
  "hardware_key_tag": "QW12DylRTmF89iGkpydNDWW7m8bVpa2Fn9KBeXGYtfX",
  "cnf": {
    "jwk": {
      "crv": "P-256",
      "kty": "EC",
      "x": "8FJtI-yr3pjyRKGMnz4WmdnQD_uJSq4R95Nj98b44",
      "y": "MKZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg"
    }
  },
  "keys_to_attest": [
    "eyJ0eXAiOiJrZXktYXR0ZXN0YXRpb24tcmVxdWVzdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6Ik9LSEhrVk5Pck
thUFZKdWZsREt3MVNRSEZOWTVpeTlPaXdBdHBBMGNvSUEifQ.eyJ3c2NkX2tleV9hdHRlc3RhdGlvbiI6eyJzdG9yYWdlX3
R5cGUiOiJMT0NBTF9OQVRJVkUifSwiY25mIjp7Imp3ayI6eyJrdHkiOiJFQyIsIngiOiJ4QUg5U05mYXE5SjVkbWt6WFlRT
GVrNVlmcFBjOGlfUHBNUlQzMTVoak1rIiwieSI6IlBFMlhMY3BXNmVYSDRGbFlHTlA5Qmh3UVFkRWlaRTF0QWRULUVpaEFD
QzgiLCJjcnYiOiJQLTI1NiIsImtpZCI6Ik9LSEhrVk5PckthUFZKdWZsREt3MVNRSEZOWTVpeTlPaXdBdHBBMGNvSUEifX0
sImlhdCI6MTc3MzA1Mzg2MSwiZXhwIjoxNzczMDU3NDYxfQ.Rn3D0GwYYZJaupzJ6617V0xav_HH6bGnttoGrD4lwY8ICPH
9NiEbTF9ZBYD3aHh20Z9GCjQ8Fhit5Fbps8v9Aw",
    "eyJ0eXAiOiJrZXktYXR0ZXN0YXRpb24tcmVxdWVzdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6IkViUUJSQ2dLNW
JrVzlZNU1idGEwZlpzMVdhVTBLZVpiek9iTXVvY2NLb28ifQ.eyJ3c2NkX2tleV9hdHRlc3RhdGlvbiI6eyJzdG9yYWdlX3
R5cGUiOiJMT0NBTF9OQVRJVkUifSwiY25mIjp7Imp3ayI6eyJrdHkiOiJFQyIsIngiOiJEVVFWTGhLMUtRUmQtZ3g3UU5jY
VNhWENnOXg0S3R6QmstNWIxWTNkeWU0IiwieSI6IkZxVjk0TWVrVm5fQ05mNTIxdm1vLVFIcWZOb k12eGdIR3NFeDlCTlc4
aFEiLCJjcnYiOiJQLTI1NiIsImtpZCI6IkViUUJSQ2dLNWJrVzlZNU1idGEwZlpzMVdhVTBLZVpiek9iTXVvY2NLb28ifX0
sImlhdCI6MTc3MzA1Mzg2MSwiZXhwIjoxNzczMDU3NDYxfQ.wIYOmX8-dmuRnuaCVg1kFoTHhsvv01vbapQ8-3er-HIiAF8
19Kt3Uy0PUN_WgxP7eWMGwhkn_9tQnnhdgXLYyw"
  ],
  "platform": "iOS",
  "wallet_solution_id": "Wallet-mobile",
  "wallet_solution_version": "1.1.0"
}
```


Key Attestation Issuance Response

If the Key Attestation Issuance Request is successfully validated, the Wallet Provider returns an HTTP response with a status code of `200 OK` and `Content-Type application/json`. The returned JSON Object includes `key_attestation` (see *Key Attestation Issuance* (page 288)). `key_attestation` is signed by the Wallet Provider (*WP_027–029* (page 739) and *WP_143–144* (page 767)). The JWT formatted Key Attestation is to be used for the Issuance phase, as an `key_attestation` JOSE header in JWT `proof` type, and will be sent to the Credential Issuer as discussed in *Digital Credential Issuance* (page 419).

The JSON Object returned in the response has the following claim:

Parameter	Description	Reference
key_attestation	REQUIRED. A String representing the issued Key Attestation.	This specification.

The value of `key_attestation` parameter is a string representing the Key Attestation in a JWT.

If any errors occur during the process, an error response is returned as it is defined in the previous section.

The following table lists HTTP Status Codes and related error codes for the ones that are different from what is already reported:

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters (e.g., header parameters, Integrity Assertion, or `keys_to_attest`), or includes invalid and unknown parameters.
`403 Forbidden`	`invalid_request`	The Integrity Assertion or Key Attestation (`keys_to_attest`) validation failed; the Integrity Assertion or Key Attestation (`keys_to_attest`) is tampered with or improperly signed.
`403 Forbidden`	`invalid_request`	The signature of the Key Attestation Request is invalid or does not match the associated public key (JWK).

Key Attestation JWT

The JOSE header of the Key Attestation JWT contains the following parameters:

JOSE header	Description	Reference
alg	REQUIRED. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	**RFC 7516#section-4.1.1**.
kid	REQUIRED. Unique identifier of the public key associated to the private key the Wallet Provider used to sign the Key Attestation.	**RFC 7638#section_3**.
typ	REQUIRED. It MUST be set to `key-attestation+jwt`	OPENID4VC-HAIP.

continues on next page

Table 13.14 – continued from previous page

JOSE header	Description	Reference
trust_chain	OPTIONAL. Sequence of Entity Statements that composes the Trust Chain related to the Wallet Provider.	OID-FED Section 4.3 *Trust Chain Header Parameter*.
x5c	REQUIRED. Contains the X.509 public key certificate or certificate chain (**RFC 5280**) corresponding to the key used to digitally sign the JWT.	**RFC 7515** Section 4.1.8.

The body of the Key Attestation JWT contains the following claims:

Claim	Description	Reference
iss	REQUIRED. String containing the URL identifying the Wallet Provider.	**RFC 7519**.
exp	REQUIRED. UNIX Timestamp with the expiry time of the JWT.	**RFC 9126** and **RFC 7519**.
iat	REQUIRED. UNIX Timestamp with the time of JWT issuance.	**RFC 9126** and **RFC 7519**.
attested_keys	REQUIRED. A non-empty array of attested keys from the same key storage component using the syntax of JWK, containing the public part of an asymmetric key pair owned by the Wallet Instance.	**RFC 7517**.
key_storage	REQUIRED. A non-empty array of case sensitive strings that assert the attack potential resistance of the key storage component and its keys attested in the `attested_keys` parameter. The following values are defined as a value for this claim: • `iso_18045_high`: It MUST be used when key storage is resistant to attack with attack potential `High`. • `iso_18045_moderate`: It MUST be used when key storage is resistant to attack with attack potential `Moderate`. • `iso_18045_enhanced-basic`: It MUST be used when key storage is resistant to attack with attack potential `Enhanced-Basic`. • `iso_18045_basic`: It MUST be used when key storage is resistant to attack with attack potential `Basic`.	OpenID4VCI.

continues on next page

Table 13.15 – continued from previous page

Claim	Description	Reference
user_authentication	REQUIRED. A non-empty array of case sensitive strings that assert the attack potential resistance of the user authentication methods allowed to access the private keys from the `attested_keys` parameter. The following values are defined as a value for this claim: • `iso_18045_high`: It MUST be used when user authentication is resistant to attack with attack potential `High`. • `iso_18045_moderate`: It MUST be used when user authentication is resistant to attack with attack potential `Moderate`. • `iso_18045_enhanced-basic`: It MUST be used when user authentication is resistant to attack with attack potential `Enhanced-Basic`. • `iso_18045_basic`: It MUST be used when user authentication is resistant to attack with attack potential `Basic`.	OpenID4VCI.
status	REQUIRED. Status mechanism for the Key Attestation. • **status_list**: REQUIRED. a status list reference as specified in Appendix D of OpenID4VCI. The value represents the revocation state of the WSCD or Keystore.	EUDI-TS 3.
certification	OPTIONAL. A String that contains a URL that links to the certification of the key storage component.	OpenID4VCI.

The Wallet Provider MUST set `key_storage` and `user_authentication` to ISO 18045 values that match the attack-potential resistance of the actual Keystore or WSCD and of the User authentication method. The Wallet Provider MUST document the mapping used for TEE, StrongBox and Secure Enclave. A Local Internal Keystore MUST NOT be attested with `iso_18045_high`. A WSCD certified as resistant to attackers with high attack potential (for example Common Criteria EAL4+ AVA_VAN.5) MAY be attested with `iso_18045_high` (*WP_022a* (page 739)).

Below is a non-normative example of the Key Attestation JWT header and payload, without encoding and signature applied:

```
{
    "typ": "key-attestation+jwt",
    "x5c": [
```

```
      "MIICrDCCAlKgAwIBAgIUCQ5zD8eGxAN6c5isgQk0jkp3qzwwCgYIKoZIzj0EAwIwgZIxCzAJBgNVBAYTAklUMQ4w
DAYDVQQIDAVMYXppbzENMAsGA1UEBwwEUm9tYTEWMBQGA1UECgwNUGFnb1BBIFMucC5BLjEkMCIGA1UEAwwbZm9vMTEuYmx
vYi5jb3JlLndpbmRvd3MubmV0MSYwJAYJKoZIhvcNAQkBFhdwYWdvcGFzcGFAcGVjLnBhZ29wYS5pdDAeFw0yNTA3MDMxNT
E2NTlaFw0yNjA3MDMxNTE2NTlaMIGSMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNVBAcMBFJvbWExFjAUB
gNVBAoMDVBhZ29QQSBTLnAuQS4xJDAiBgNVBAMMG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDEmMCQGCSqGSIb3DQEJ
ARYXcGFnb3Bhc3BhQHBlYy5wYWdvcGEuaXQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARtdnuBd5hAg3cPyz7/o8vBFyL
4sp45HikFogMZse7f9/iL4gn6YM/ehP1CQI0dFnT3c/RdxFyRukKWXscopwXso4GDMIGAMAkGA1UdEwQCMAAwCwYDVR0PBA
QDAgWgMCYGA1UdEQQfMB2CG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDAdBgNVHQ4EFgQUOd7M/0bo1+saPT/H9M+G9
6sKnYIwHwYDVR0jBBgwFoAUkKHxQLaudIm0WfwZeN5HMTG8SCMwCgYIKoZIzj0EAwIDSAAwRQIgSXKs9Gx2bXVG9nxap1I/
KqwHdh6SQDKI85J9n7FqrI8CIQCqltHTw6sci1R4RSOFySEvzEohQTbNTke+X2lD9iQd5A==",
      "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0OTMzWhcNMjcwNjA0MTI0OTMzWjCBkjELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMRYwFAYDVQQKDA1QYWdvUEEgUy5wLkEuMSQwIgYDVQQDDBtmb28xMS5ibG9iLmNvcmUud2luZG93
cy5uZXQxJjAkBgkqhkiG9w0BCQEWF3BhZ29wYXNwYUBwZWMucGFnb3BhLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA
EgXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l/iX7p1ysisVV2vOhj1xEm/3M24U5HcywPPpNIOEHTESjqOCAW4wgg
FqMB0GA1UdDgQWBBSQofFAtq50ibRZ/Bl43kcxMbxIIzCB5QYDVR0jBIHdMIHagBTzUINxjECpXem0zyIpsnTqv3DEGaGBu
aSBtjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQ
b2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYDVQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHp
zLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5pcHpzLml0ggYBlzr5eUMwEgYDVR0TAQH/BAgwBgEB/wIBAD
AOBgNVHQ8BAf8EBAMCAAcwPQYDVR0fBDYwNDAyoDCgLoYsaHR0cHM6Ly9wcmUudGEud2FsbGV0LmlwenMuaXQvcGtpL3RhL
XN1Yi5jcmwwCgYIKoZIzj0EAwIDRwAwRAIgBSyH++oPDP8ipRxHcJ8u2HkN7wL44XpJFxSIBV9RFHMCIBixpcT43NSsk0Ev
hJb4g68c1d9OrBnXOXVuyCtWWE6a",
      "MIIDQzCCAuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0NTE3WhcNMzAwNjAzMTI0NTE3WjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQb2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYD
VQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHpzLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5
pcHpzLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEaE0xyhd3e9LDT7uwHOclL5H3389gwiCwFhI3KOvidn0glBIHYx
qH+4Z9VTMYWG5L8cwC9AaJUCNGu2dp5ZiiTKOB5TCB4jAdBgNVHQ4EFgQU81CDcYxAqV3ptM8iKbJ06r9wxBkwHwYDVR0jB
BgwFoAU81CDcYxAqV3ptM8iKbJ06r9wxBkwDwYDVR0TAQH/BAUwAwEB/zBEBggrBgEFBQcBAQQ4MDYwNAYIKwYBBQUHMAKG
KGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jZXIwDgYDVR0PAQH/BAQDAgEGMDkGA1UdHwQyMDAwLqA
soCqGKGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jcmwwCgYIKoZIzj0EAwIDSQAwRgIhAOsQYzR+eG
f4je63VGHqkpmkBbfyOre+mfIdHHowWWR/AiEA58xBNb5UW5uMB+tQur8fq24RD5MmRHLYS6bDgIYmluw="
    ],
    "alg": "ES256",
    "kid": "8UkfrvttLkpAQOOp"
  }

{
    "attested_keys": [
        {
            "crv": "P-256",
            "kty": "EC",
            "x": "4HNptI-xr2pjyRJKGMnz4WmdnQD_uJSq4R95Nj98b44",
```

```
            "y": "LIZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg",
            "kid": "vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"
        }
    ],
    "iss": "https://wallet-provider.example.org",
    "key_storage": [ "iso_18045_moderate" ],
    "status": {
        "status_list": {
            "idx": 412,
            "uri": "https://revocation_url/statuslists/1"
        }
    },
    "user_authentication": [ "iso_18045_moderate" ],
    "iat": 1687281195,
    "exp": 1687288395
}
```


Note

As the certification scheme has not yet been defined, the exact content of `certification` is undefined and will be specified in a future update.

Note

As a revocation mechanism for KA, the Type-shared index described in Section 2.5.2 of EUDI-TS 3 is preferred.

13.1.6 e-Service PDND Wallet Provider Catalog

User's death leads to the revocation of the Wallet Instances of the User and the deletion of the User account at the Wallet Provider. For this reason, the Wallet Provider provides the following e-service through PDND. A PID/IT-Wallet ID Provider that has been notified by the Authentic Source of the PID/IT-Wallet ID of the User's death MUST send a notification to Wallet Providers using this endpoint. Calls to the e-Service PDND Wallet Provider Catalogue, including Notify User Death, MUST be authenticated and authorised. Unauthenticated requests MUST be rejected. A successful Notify User Death MUST lead to Wallet Instance revocation as for other PID/IT-Wallet ID Provider-initiated revocations (*WP_161* (page 739)).

Note

A complete OpenAPI Specification is available *Wallet Provider PDND OpenAPI Specification* (page 823).

Notify User Death

Description	This service is used to notify the Wallet Provider of the need to revoke the Wallet Instance and delete the User's account due to the User's death.
Provider	Wallet Provider
Consumer	PID Provider

13.2 Credential Issuer Endpoints

13.2.1 Metadata Endpoints

The Credential Issuers MUST provide an Entity Configuration through the `/.well-known/openid-federation` endpoint, according to Section *Entity Configuration* (page 117). Technical details are provided in Section credential-issuer-entity-configuration:Credential Issuer Entity Configuration.

Alternatively the Credential Issuer's Metadata can be retrieved using the Credential Issuer Identifier. The Metadata document MUST be made available in JSON and JWT format through the `/.well-known/openid-credential-issuer` endpoint as defined in Section 12.2.2 of OpenID4VCI.

The `Accept-Language` header in the HTTP GET request can be used to indicate the language(s) preferred. In this case the Credential Issuer can send a subset of the metadata containing internationalized display data for one or all of the requested languages and indicate returned languages using the HTTP `Content-Language` Header.

Below is a non-normative example.

```
GET /.well-known/openid-credential-issuer HTTP/1.1
Host: issuer.example.com
Accept: application/json
Accept-Language: it-IT, it;q=0.9
```

```
GET /.well-known/openid-credential-issuer HTTP/1.1
Host: issuer.example.com
Accept: application/jwt
Accept-Language: it-IT, it;q=0.9
```

The Credential Issuer MUST respond with HTTP Status Code 200 and return the Credential Issuer Metadata containing:

- the parameters defined in credential-issuer-metadata:Metadata for openid_credential_issuer as an unsigned JSON document using the media type *application/json*.

  or

- the header and payload parameters defined in Section 12.2.3 of OpenID4VCI in addition to those defined in credential-issuer-metadata:Metadata for openid_credential_issuer as a signed JSON Web Token using the media type *application/jwt*.

Below is a non-normative example of Credential Issuer metadata in signed form:

ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgInR5cCI6ICJvcGVuaWR2Y2ktaXNzdWVyLW1ldGFkYXRhK2p3dCIsDQogICJraW
QiOiAiZEI2N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIgDQp9.ew0KCSJpYXQiOiAxNzE4MjA
3MjE3LA0KCSJleHAiOiAxNzQ5NzQzMjE2LA0KCSJpc3MiOiAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmciL
A0KCSJzdWIiOiAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmciLA0KCSJjcmVkZW50aWFsX2lzc3VlciI6ICJ
odHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZyIsDQoJImNyZWRlbnRpYWxfZW5kcG9pbnQiOiAiaHR0cHM6Ly9lY
WEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvY3JlZGVudGlhbCIsDQoJIm5vbmNlX2VuZHBvaW50IjogImh0dHBzOi8vZWFhLXB
yb3ZpZGVyLmV4YW1wbGUub3JnL25vbmNlLWVuZHBvaW50IiwNCgkiZGVmZXJyZWRfY3JlZGVudGlhbF9lbmRwb2ludCI6I
CJodHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZy9kZWZlcnJlZC1jcmVkZW50aWFsIiwNCgkibm90aWZpY2F0aW9
uX2VuZHBvaW50IjogImh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnL25vdGlmaWNhdGlvbiIsDQoJImRpc3BsY
XkiOiBbDQoJCXsNCgkJCSJuYW1lIjogIkVBQSBQcm92aWRlciIsDQoJCQkibG9jYWxlIjogIml0LUlUIiwNCgkJCSJsb2d
vIjogew0KCQkJCSJ1cmki0iAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvcHVibGljL2NvbXBhY
3QtbG9nby1VUy5zdmciLA0KCQkJCSJ1cmkjaW50ZWdyaXR5IjogIjSbHR5Z283YktCRTRFNDk4dGIvbExuOURJa2ltQmhyenJzVUV
SeVVRam89IiwNCgkJCQkiYWx0X3RleHQiOiAiVGVzdG8gYWx0ZXJuYXRpdm8gYWxsJ2ltbWFnaW5lIGRlbCBsb2dvIg0KC
QkJfQ0KDQoJCX0sDQoJCXsNCgkJCSJuYW1lIjogIkVBQSBQcm92aWRlciIsDQoJCQkibG9jYWxlIjogImVuLVVTIiwNCgk
JCSJsb2dvIjogew0KCQkJCSJ1cmki0iAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvcHVibGljL2NvbXBhY
3QtbG9nby1VUy5zdmciLA0KCQkJCSJ1cmkjaW50ZWdyaXR5IjogIm1CUUVpNDJhajBWWW5GbmdwT2picUpHa3VDbkI1TG1
iWlJuV3BPK0FIcVVCVlF4T2U5a01JUE96K3BTalJBckUiLA0KCQkJCSJhbHRfdGV4dCI6ICJBbHRlcm5hdGl2ZSB0ZXh0I
GZvciB0aGUgbG9nbyBpbWFnZSINCgkJCX0NCgkJfQ0KCV0sDQoJImNyZWRlbnRpYWxfY29uZmlndXJhdGlvbnNfc3VwcG9
ydGVkIjogew0KCQkidm50IjogInVybjpldWRpOkV1cm9wZWFuRGlzYWJpbGl0eUNhcmQiOiB7DQoJCQkiZm2Rfand0X0V1cm9wZWFuRGlzYWJpbGl0eUNhcmQiOiB7DQoJCQkJ
Wp3dCIsDQoJCQkic2NvcGUiOiAiRXVyb3BlYW5EaXNhYmlsaXR5Q2FyZCIsDQoJCQkiY3J5cHRvZ3JhcGhpY19iaW5kaW5
nX21ldGhvZHNfc3VwcG9ydGVkIjogWw0KCQkJqd2siDQoJCQldLA0KCQkJImNyZWRlbnRpYWxfc2lnbmluZ19hbGdfd
mFsdWVzX3N1cHBvcnRlZCI6IFsNCgkJCUFUzM4NCIsDQoJCQkJIkVTNTEyIg0KCQkJXSwNCgkJCSJwcm9vZl90eXBlc19zdXBwb3J0ZW
JSwcm9vZl90eXBlc19zdXBwb3J0ZWQiOiB7DQoJCQkJImp3dCI6IHsNCgkJCQkJInByb29mX3NpZ25pbmdfYWxnX3Zhb
HVlc19zdXBwb3J0ZWQiOiBbDQoJCQkJCSJIkVTMzg0IiwNCgkJCQkJSJFUzUxMiINCgkJCQ
JXSwNCgkJCQJImtleV9hdHRlc3RhdGlvbnNfcmVxdWlyZWQiOiB7DQoJCQkJrZXlfc3RvcmFnZSI6IFsgImlzb18xO
DA0NV9lbmhhbmNlZC1iYXNpYyIgXSwNCgkJCQJInVzZXJfYXV0aVudGljYXRpb24iOiBbICJpc29fMTgwNDVfZW5oYW5
jZWQtYmFzaWMiIF0NCgkJCQJImtleV9hdHRlc3RhdGlvbnNfcmVxdWlyZWQiOiB7DQoJCQkJrZXlfc3RvcmFnZSI6IFsgImlzb18xO
DA0NV9lbmhhbmNlZC1iYXNpYyIgXSwNCgkJCQJInVzZXJfYXV0aGVudGljYXRpb24iOiBbICJpc29fMTgwNDVfZW5oYW5
jZWQtYmFzaWMiIF0NCgkJCQJImtleV9hdHRlc3RhdGlvbnNfcmVxdWlyZWQiOiB7DQoJCQkJImp3dCI6IHsNCgkJCQkJ
WxlIjogImVuLVVTIiwNCgkJCQJSJkZXNjcmlwdGlvbiI6ICJEaWdpdGFsIHZlcnNpb24gb2YgdGhlIEV1cm9wZWFuIER

The `authorization_servers` entries of the Credential Issuer Metadata can be used to obtain the Authorization Server metadata from the Oauth Authorization Server `/.well-known/oauth-authorization-server` as defined in Section 3 of **RFC 8414**. In case the `authorization_servers` parameter is omitted, the Credential Issuer's identifier can be used to retrieve the Authorization Server metadata.

Below is a non-normative example.

```
GET /.well-known/oauth-authorization-server HTTP/1.1
Host: oauth-authorization-server.example.com
```

The Oauth Authorization Server MUST respond with HTTP Status Code 200 and return the Oauth Authorization Server Metadata containing the parameters defined in credential-issuer-metadata:Metadata for oauth_authorization_server, using a JSON document with the media type *application/json*.

13.2.2 Credential Issuance Endpoints

Pushed Authorization Request Endpoint

Pushed Authorization Request (PAR) Request

The request to the Credential Issuer authorization endpoint MUST use HTTP Headers parameters and HTTP POST parameters.

The HTTP POST method MUST use the parameters in the message body encoded in `application/x-www-form-urlencoded` format.

Table 13.17: PAR http request parameters

Claim	Description	Reference
client_id	MUST be set to the thumbprint of the `jwk` value in the `cnf` parameter inside the Wallet Instance Attestation.	**RFC 6749**
request	It MUST be a signed JWT. The private key corresponding to the public one in the `cnf` parameter inside the Wallet Instance Attestation MUST be used for signing the Request Object.	OpenID Connect Core. Section 6

The Pushed Authorization Endpoint is protected with OAuth 2.0 Attestation-based Client Authentication [OAUTH-ATTESTATION-CLIENT-AUTH], therefore the request to the Credential Issuer authorization endpoint MUST use the following HTTP Headers parameters:

Table 13.18: http request header parameters

OAuth-Client-Attestation	It MUST be set to a value containing the Wallet Instance Attestation JWT.	OAUTH-ATTESTATION-CLIENT-AUTH.
OAuth-Client-Attestation-PoP	It MUST be set to a value containing the Wallet Instance Attestation JWT Proof of Possession.	OAUTH-ATTESTATION-CLIENT-AUTH.

Note

Clients SHOULD select the algorithms for the Wallet Instance Attestation and its proof of possession according to the Authorization Server metadata fields `client_attestation_signing_alg_values_supported` and `client_attestation_pop_signing_alg_values_supported` documented in credential-issuer-metadata:Metadata for oauth_authorization_server.

The JWT *Request Object* has the following JOSE header parameters:

JOSE header	Description	Reference
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	**RFC 7516#section-4.1.1**.
kid	Unique identifier of the `jwk` inside the `cnf` claim of Wallet Instance Attestation as base64url-encoded JWK Thumbprint value.	**RFC 7638#section_3**.
typ	REQUIRED. Media Type of the JWT. It MUST be set to `oauth-authz-req+jwt` to explicitly declare that this JWT is an Authorization Request Object and to prevent JWT confusion attacks.	**RFC 9101** and **RFC 8725**.

The `request` JWT payload contained in the HTTP POST message is given with the following parameters:

Claim	Description	Reference
iss	It MUST be set to the `client_id`.	**RFC 9126** and **RFC 7519**.
aud	It MUST be set to the identifier of the Credential Issuer.	**RFC 9126** and **RFC 7519**.

continues on next page

Table 13.20 – continued from previous page

Claim	Description	Reference
exp	UNIX Timestamp with the expiry time of the JWT. The claim value MUST be not greater than 300 seconds from the issuance time.	**RFC 9126** and **RFC 7519**.
iat	UNIX Timestamp with the time of JWT issuance.	**RFC 9126** and **RFC 7519**.
response_type	MUST be set to `code`.	**RFC 6749**
client_id	It MUST be set as in the *Table of the HTTP parameters* (page 565).	See *Table of the HTTP parameters* (page 565).
state	Unique session identifier at the client side. This value will be returned to the client in the response, at the end of the authentication. It MUST be a random string composed by alphanumeric characters and with a minimum length of 32 digits. Special characters MUST be considered non-alphanumeric characters as defined in [NIST].	See [OIDC] Section 3.1.2.1.
code_challenge	A challenge derived from the **code verifier** that is sent in the authorization request.	**RFC 7636# section-4.2**.
code_challenge_method	A method that was used to derive **code challenge**. It MUST be set to `S256`.	**RFC 7636# section-4.3**.
scope	REQUIRED. JSON String. String specifying a unique identifier of the Credential regardless of its format, as required by [OPENID4VC-HAIP]. It MUST be mapped in the *credential_configurations_supported* metadata claim of the Credential Issuer. Unique identifier value MUST match the *credential_type* parameter of the *Digital Credentials Catalog* (page 179). For instance, in the case of the PID, it may be set to `pid`, for IT-Wallet ID it may be set to `eid` while in case of mobile driving licence `mDL`. Since it may be multivalued, when this occurs each value MUST be separated by a space. The Wallet Instance MUST use the `scope` value advertised for the requested Credential Configuration, obtained from the Credential Issuer Metadata or, in the issuer-initiated flow, from the Credential Offer.	**RFC 6749** and [OPENID4VC-HAIP].

continues on next page

Table 13.20 – continued from previous page

Claim	Description	Reference
authorization_details	REQUIRED. Array of JSON Objects. The array MUST contain at least one JSON Object of type `openid_credential`. Each JSON Object of type `openid_credential` MUST include the following claims: • **type**: it MUST be set to `openid_credential`, • **credential_configuration_id**: JSON String. String specifying a unique identifier of the Credential in a specific format that MUST be mapped in the *credential_configurations_supported* metadata claim of the Credential Issuer. For instance,"dc_sd_jwt_pid" can be used for PID in SD-JWT VC format, `dc_sd_jwt_eid` for IT-Wallet ID in SD-JWT VC format, `dc_sd_jwt_mDL` for mobile driving licence in SD-JWT VC format and `mso_mdoc_mDL` for mobile driving license in mdoc format. When eID Substantial Authentication with MRTD Verification is indicated as a hint, according to credential-issuance-endpoint:User Authentication Method Selection, an additional JSON Object is included with the following claims: • **type**: REQUIRED. MUST be `it_l2+document_proof`, • **idphinting**: OPTIONAL. URL of the Identity Provider to be used as an optional hint for the Authorization Server, according to credential-issuance-endpoint:User Authentication Method Selection, • **challenge_method**: REQUIRED. Specifies the MRTD verification method. The value MUST be `mrtd+ias`. Additional verification methods MAY be defined in future releases of this Specification, • **challenge_redirect_uri**: REQUIRED. Redirect URI, recognized by Wallet Instance, for handling the challenge response.	See [RAR **RFC 9396**], [Open ID4VCI] and *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457).
redirect_uri	Redirection URI to which the response is intended to be sent. It MUST be a universal or app link registered with the local operating system, so this latter will provide the response to the Wallet Instance.	See [OIDC] Section 3.1.2.1.
jti	Unique identifier of the JWT that, together with the value contained in the `iss` claim, prevents the reuse of the JWT (replay attack). Since the *jti* value alone is not collision resistant, it MUST be identified uniquely together with its issuer.	[**RFC 7519**].

continues on next page

Table 13.20 – continued from previous page

Claim	Description	Reference
issuer_state	It MUST be present only in case of issuer initiated flow. It MUST contain the same value contained in the Credential Offer.	[**RFC 7519**].

Note

The Pushed Authorization Request MUST include both the *scope* parameter and the *authorization_details* parameter. The Credential Issuer MUST interpret these individually. However, if both request the same Credential type, then the Credential Issuer MUST follow the request as given by the authorization details object.

The JOSE header of the Wallet Instance Attestation proof of possession, contained in the HTTP Request headers, MUST contain:

JOSE header	Description	Reference
typ	It MUST be set to `oauth-client-attestation-pop+jwt`	OAUTH-ATTESTATION-CLIENT-AUTH.
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms listed in the Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or any symmetric algorithm (MAC) identifier.	**RFC 7516# section-4.1.1**.

The body of the Wallet Instance Attestation proof of possession JWT, contained in the HTTP Request headers, contains:

Claim	Description	Reference
iss	REQUIRED. Thumbprint of the JWK in the `cnf` parameter.	**RFC 9126** and **RFC 7519**.
aud	REQUIRED. It MUST be set to the identifier of the Credential Issuer.	**RFC 9126** and **RFC 7519**.
iat	REQUIRED. UNIX Timestamp with the time of JWT issuance.	**RFC 9126** and **RFC 7519**.
jti	REQUIRED. Unique identifier for the DPoP proof JWT. The value SHOULD be set using a *UUID v4* value according to [**RFC 4122**].	[**RFC 7519**. Section 4.1.7].

continues on next page

Table 13.22 – continued from previous page

Claim	Description	Reference
nbf	OPTIONAL. UNIX Timestamp with the start time of validity of the JWT issuance.	RFC 9126 and RFC 7519.

Pushed Authorization Request (PAR) Response

If the verification is successful, the Credential Issuer MUST provide the response with a *201 HTTP status code*. The following parameters are included as top-level members in the HTTP response message body, using the `application/json` media type as defined in [RFC 8259].

Claim	Description	Reference
request_uri	The request URI corresponding to the authorization request posted. This URI MUST be a single-use reference to the respective authorization request. It MUST contain some part generated using a cryptographically strong pseudorandom algorithm. The value format MUST be `urn:ietf:params:oauth:request_uri:<reference-value>` with `<reference-value>` as the random part of the URI that references the respective authorization request data.	[RFC 9126].
expires_in	A JSON number that represents the lifetime of the request URI in seconds as a positive integer.	[RFC 9126].

If any errors occur during the PAR Request, the Authorization Server MUST return an error response as defined in RFC 9126#section-2.3. The response MUST use *application/json* as the content type and MUST include the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of an error response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_request",
    "error_description": "The redirect_uri is not valid for the given client"
}
```

In the following table are listed HTTP Status Codes and related error codes that are supported for the error response:

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_request`	The Credential Issuer cannot fulfill the request because of missing parameters, invalid parameters or request malformed. (**RFC 6749#section-5.2**).
400 Bad Request [REQUIRED]	`invalid_scope`	The Credential Issuer cannot fulfill the request because the requested scope is invalid or unknown. (**RFC 6749#section-5.2**).
400 Bad Request [REQUIRED]	`use_fresh_attestation`	The Wallet Instance Attestation JWT is not fresh enough to be acceptable by the server. Section 6.2 of OAUTH-ATTESTATION-CLIENT-AUTH.
401 Unauthorized [REQUIRED]	`invalid_client`	The Credential Issuer cannot fulfill the request because of Client Authentication failed (for example in case of unknown client, no parameters Client Authentication included, or unsupported authentication method). (**RFC 6749#section-5.2**).
405 Method not allowed [OPTIONAL]	-	The Credential Issuer cannot fulfill the request because POST method was not used in the request. (**RFC 9126#section-2.3**).
413 Payload Too Large [OPTIONAL]	-	The Credential Issuer cannot fulfill the request because of the size of the request is higher than permitted limit.(**RFC 9126#section-2.3**).
429 Too Many Requests [OPTIONAL]	-	The Credential Issuer cannot fulfill the request because of the numbers requests received is higher than permitted limit. (**RFC 9126#section-2.3**).
500 Internal Server Error [REQUIRED]	`server_error`	The Credential Issuer encountered an internal problem. (**RFC 6749#section-4.1.2.1**).
503 Service Unavailable [REQUIRED]	`temporarily_unavailable`	The Credential Issuer is temporary unavailable. (**RFC 6749#section-4.1.2.1**).
504 Gateway Timeout [OPTIONAL]	-	The Credential Issuer cannot fulfill the request within the defined time interval.

Authorization endpoint

The authorization endpoint is used to interact with the Credential Issuer and obtain an authorization grant. The authorization server MUST first verify the identity of the User that own the credential.

User Authentication Method Selection

The Authorization Server of the Credential Issuer is responsible for authenticating the User according to its authentication policies and the applicable Trust Framework. The decision on which authentication method to apply, and whether to present a discovery page of the supported methods, MUST remain with the Authorization Server of the Credential Issuer.

For PID issuance:

- The Authorization Server MUST authenticate the User using CieID with Level of Assurance High (CIE L3).

For IT-Wallet ID issuance:

- The Authorization Server MUST authenticate the User using one of the methods it supports, including CieID with Level of Assurance High (CIE L3) and, if supported, eID Substantial Authentication with MRTD Verification as defined in *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457).
- The Authorization Server MAY present a discovery page enabling the User to select among the authentication methods it supports and allows for the request in progress.
- The Wallet Instance MAY include in the Pushed Authorization Request one or more optional hints related to the User's preferred authentication method, as defined below. Such hints MUST NOT be interpreted as a binding imposition of the authentication method by the Wallet Instance but are essential recommendations to follow in order to not compromise the User Experience.
- When the Wallet Instance indicates eID Substantial Authentication with MRTD Verification, it MUST include in the `authorization_details` array an object with `type` set to `it_l2+document_proof`, with the structure and claims defined in the Table of the JWT Request parameters of Section credential-issuance-endpoint:Pushed Authorization Request Endpoint.
- If the Authorization Server receives a hint that it supports and accepts, SHOULD omit presenting a discovery page and/or autonomously applying only its default authentication method.
- If the hint is absent, unsupported, or not accepted, the Authorization Server MUST apply its authentication policies, including any discovery page and the default authentication method.
- The Authorization Server MUST NOT offer on a discovery page, select, or otherwise apply eID Substantial Authentication with MRTD Verification unless the Pushed Authorization Request includes an `it_l2+document_proof` object containing the required technical parameters `challenge_method` and `challenge_redirect_uri`. Without those parameters the

Authorization Server cannot complete the subsequent multi-step flow and browser confirmation.

- A Wallet Instance MUST NOT be required to implement an authentication-method discovery interface as an interoperability prerequisite for IT-Wallet ID issuance. Any Wallet-side pre-selection interfaces (for example to improve User experience) remain optional and MUST be limited to providing hints to the Authorization Server.

The authentication hints recognized for IT-Wallet ID issuance are the following:

- the presence in the `authorization_details` array of an object with `type` set to `it_l2+document_proof` constitutes an optional hint by which the Wallet Instance indicates a preference for eID Substantial Authentication with MRTD Verification;
- the `idphinting` claim within that object, when present, constitutes an optional hint for the URL of the LoA3 Identity Provider to be used in the context of that method;
- the absence of the `it_l2+document_proof` object does not, by itself, constitute a Wallet Instance mandate on the authentication method: in that case the Authorization Server applies its policies, and MUST NOT offer or apply eID Substantial Authentication with MRTD Verification for that request. In the absence of a User selection via a Credential Issuer discovery page, the IT-Wallet ID Provider default authentication method MUST be CieID with Level of Assurance High (CIE L3).

Note

This Section concerns authentication method selection during PID and IT-Wallet ID issuance and does not apply to the Relying Party Touchpoint Discovery Page described in *User Experience Design* (page 35), which has a distinct purpose (selection of the method to access a service).

For (Q)EAA issuance, the Authorization Server MUST authenticate the User by requesting presentation of a valid PID from the Wallet Instance using [OpenID4VP], as described in *Low-Level Issuance Flow* (page 422). The Credential Issuer MAY instead request presentation of a valid IT-Wallet ID, according to its authentication policies. In that context, discovery of SPID, CIE L3, or L2+ methods does not apply.

Authorization Request

The Authorization request is issued by the Web Browser in use by the Wallet Instance, the HTTP methods **POST** or **GET** are used. When the method **POST** is used, the parameters MUST be sent using the *Form Serialization*. When the method **GET** is used, the parameters MUST be sent using the *Query String Serialization*. For more details see Section 13 of [OIDC].

The mandatory parameters in the HTTP authentication request are specified in the following table.

Claim	Description	Reference
client_id	It MUST be set as in the *Table of the HTTP parameters* (page 565).	See *Table of the HTTP parameters* (page 565).
request_uri	It MUST be set to the same value as obtained by PAR Response. See *Table of the HTTP PAR Response parameters* (page 570).	[**RFC 9126**].

Authorization Response

The authentication response is returned by the Credential issuer authorization endpoint at the end of the authentication flow.

If the authentication is successful the Credential Issuer redirects the User by adding the following query parameters as required to the *redirect_uri*. The redirect URI MUST be a universal or app link registered with the local operating system, so this latter is able to provide the response to the Wallet Instance.

Claim	Description	Reference
code	Unique *Authorization Code* that the Wallet Instance submits to the Token Endpoint.	[**RFC 6749#section-4.1.2**], [**RFC 7521**].
state	The Wallet Instance MUST check the correspondence with the `state` parameter value in the Request Object. It is defined as in the *Table of the JWT Request parameters* (page 566).	[**RFC 6749#section-4.1.2**].
iss	Unique identifier of the Credential Issuer who created the Authentication Response. The Wallet Instance MUST validate this parameter.	[**RFC 9207**], [**RFC 7519**, Section 4.1.1.].

If any errors occur during the Authorization Request, the Authorization Server MUST return an error response as defined in **RFC 6749#section-4.1.2.1**. In case of invalid/missing `redirect_uri` or `client_id` Authorization Server MUST inform the User with the error and MUST NOT

redirect the User to the redirection URI. If any other error occurs the Authorization Server MUST redirect the User by adding the following query parameters as required to the *redirect_uri* using the *application/x-www-form-urlencoded* format:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.
- *state*. The exact value of `state` parameter contained in the Request Object.

Below is a non-normative example of an error response.

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?
 error=invalid_request
 &error_description=Unsupported%20response_type%20value
 &state=fyZiOL9Lf2CeKuNT2JzxiLRDink0uPcd
```

In case of Authorization Server redirects the User to the *redirect_uri* HTTP status code *302 (Found)* MUST be used. The following error codes are supported for the error response:

Status Code	error code	Description
302 Found [REQUIRED]	`invalid_request`	The Credential Issuer cannot fulfill the request because of missing parameters, invalid parameters or request malformed. (**RFC 6749#section-4.1.2.1**).
302 Found [REQUIRED]	`unauthorized_client`	The Credential Issuer cannot fulfill the request because the client is not authorized to request an authorization code. (**RFC 6749#section-4.1.2.1**).
302 Found [REQUIRED]	`server_error`	The Credential Issuer encountered an internal problem. (**RFC 6749#section-4.1.2.1**).
302 Found [REQUIRED]	`temporarily_unavailable`	The Credential Issuer is temporary unavailable. (**RFC 6749#section-4.1.2.1**).

In case of Authorization Server doesn't redirect the User to the *redirect_uri* the following HTTP Status Codes are supported for the error response:

Status Code	Description
400 Bad Request [REQUIRED]	The Credential Issuer cannot fulfill the request because of invalid/missing `redirect_uri` or `client_id` parameter.
500 Internal Server Error [REQUIRED]	The Credential Issuer encountered an internal problem.

continues on next page

Table 13.28 – continued from previous page

Status Code	Description
503 Service Unavailable [REQUIRED]	The Credential Issuer is temporary unavailable.
504 Gateway Timeout [OPTIONAL]	The Credential Issuer cannot fulfill the request within the defined time interval.

Token endpoint

The token endpoint is used by the Wallet Instance to obtain an Access Token by presenting an authorization grant, as defined in **RFC 6749**. The Token Endpoint is a protected endpoint with a client authentication based on the model defined in OAuth 2.0 Attestation-based Client Authentication [OAUTH-ATTESTATION-CLIENT-AUTH].

Token Request

The request to the Credential Issuer Token endpoint MUST be an HTTP request with method POST, with the body message encoded in `application/x-www-form-urlencoded` format. The Wallet Instance sends the Token endpoint request with `OAuth-Client-Attestation` and `OAuth-Client-Attestation-PoP` as header parameters according to OAUTH-ATTESTATION-CLIENT-AUTH.

The Token endpoint is protected with *OAuth 2.0 Attestation-based Client Authentication* [OAUTH-ATTESTATION-CLIENT-AUTH], therefore the request to the Credential Issuer authorization endpoint MUST use the following HTTP Headers parameters **OAuth-Client-Attestation** as **OAuth-Client-Attestation-PoP** as defined in the credential-issuance-endpoint:Pushed Authorization Request Endpoint"

The Token endpoint issues DPoP tokens, therefore it is REQUIRED that the request includes in its HTTP header the DPoP proof parameter. The Authorization Server MUST validate the DPoP proof received at the Token endpoint, according to **RFC 9449** Section 4.3. This mitigates the misuse of leaked or stolen Access Tokens/Refresh Tokens at the Credential/Token endpoint. If the DPoP proof is invalid, the Token endpoint returns an error response, according to Section 5.2 of [**RFC 6749**] with `invalid_dpop_proof` as the value of the error parameter.

The token request contains the following claims:

Claim	Description	Reference
grant_type	REQUIRED. It MUST be set to `authorization_code` or `refresh_token`.	[**RFC 6749**].
code	REQUIRED only if the grant type is `authorization_code`. Authorization code returned in the Authentication Response. It MUST NOT be present if grant type is `refresh_token`.	[**RFC 6749**].
redirect_uri	REQUIRED only if the grant type is `authorization_code`. It MUST be set as in the Request Object *Table of the JWT Request parameters* (page 566). It MUST NOT be present if grant type is `refresh_token`.	[**RFC 67491**].
code_verifier	REQUIRED only if the grant type is `authorization_code`. Verification code of the **code_challenge**.	Proof Key for Code Exchange by OAuth Public Clients. It MUST NOT be present if grant type is `refresh_token`.
refresh_token	REQUIRED only if the grant type is `refresh_token`. The Refresh Token previously issued to the Wallet Instance. It MUST NOT be present if grant type is `authorization_code`.	[**RFC 6749**].
scope	OPTIONAL only if the grant type is `refresh_token`. The requested scope MUST NOT include any scope not originally granted by the User, and if omitted is treated as equal to the scope originally granted by the User. It MUST NOT be present if grant type is `authorization_code`.	[**RFC 6749**].

A **DPoP Proof JWT** is included in the HTTP request using the `DPoP` header parameter containing a DPoP JWT.

The JOSE header of a **DPoP JWT** MUST contain at least the following parameters:

JOSE header	Description	Reference
typ	It MUST be equal to `dpop+jwt`.	[**RFC 7515**] and [**RFC 8725**. Section 3.11].
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or with a symmetric algorithm (MAC) identifier.	[**RFC 7515**].

continues on next page

Table 13.30 – continued from previous page

JOSE header	Description	Reference
jwk	It represents the public key chosen by the Wallet Instance, in JSON Web Key (JWK) [**RFC 7517**] format that the Access Token MUST be bound to, as defined in [**RFC 7515**] Section 4.1.3. It MUST NOT contain a private key.	[**RFC 7517**] and [**RFC 7515**].

The payload of a **DPoP JWT Proof** MUST contain the following claims:

Claim	Description	Reference
jti	Unique identifier for the DPoP proof JWT. The value SHOULD be set using a *UUID v4* value according to [**RFC 4122**].	[**RFC 7519**. Section 4.1.7].
htm	The value of the HTTP method of the request to which the JWT is attached.	[**RFC 9110**. Section 9.1].
htu	The HTTP target URI, without query and fragment parts, of the request to which the JWT is attached.	[**RFC 9110**. Section 7.1].
iat	UNIX Timestamp with the time of JWT issuance, coded as NumericDate as indicated in **RFC 7519**.	[**RFC 7519**. Section 4.1.6].

Token Response

If the Token Request is successfully validated, the Authorization Server provides an HTTP Token Response with a *200 (OK)* status code. The Token Response contains the following claims.

Claim	Description	Reference
access_token	REQUIRED. The *DPoP-bound Access Token*, in signed JWT format, allows accessing the Credential Endpoint for obtaining the Credential.	[**RFC 6749**].
refresh_token	OPTIONAL. The *DPoP-bound Refresh Token*, in signed JWT format, which can be used to obtain a new Access Token at the Credential Issuer Token Endpoint.	[**RFC 6749**].
token_type	REQUIRED. Type of *Access Token* returned. It MUST be equal to `DPoP`.	[**RFC 6749**].
expires_in	REQUIRED. Expiry time of the *Access Token* in seconds.	[**RFC 6749**].

continues on next page

Table 13.32 – continued from previous page

Claim	Description	Reference
authorization_details	REQUIRED. Array of JSON Objects, used to identify Credentials with the same metadata but different claimset/claim values and/or simplify the Credential request even when only one Credential is being issued. In addition to the claim defined in *Table of the JWT Request parameters* (page 566) it MUST include the following claim: • **credential_identifiers**: Array of strings, each uniquely identifying a Credential dataset that is available for the issuance.	[OpenID4VCI].

If any errors occur during the validation of the Token Request, the Authorization Server MUST return an error response as defined in **RFC 6749#section-5.2**. The response MUST use the HTTP Content-Type set to *application/json* and MUST include the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of an error response.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "error": "invalid_client",
    "error_description": "Client authentication failed"
}
```

In the following table are listed HTTP Status Codes and related error codes that are supported for the error response:

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_request`	The Credential Issuer cannot fulfill the request because of missing parameters, invalid parameters or request malformed. (**RFC 6749#section-5.2**).

continues on next page

Table 13.33 – continued from previous page

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_grant`	The Credential Issuer cannot fulfill the request because the provided authorization code or Refresh Token is invalid, expired, revoked, or does not match the redirection URI used in the authorization request, or was issued to another client. (**RFC 6749#section-5.2**).
400 Bad Request [REQUIRED]	`unsupported_grant_type`	The Credential Issuer cannot fulfill the request because the authorization grant type is not supported. (**RFC 6749#section-5.2**).
400 Bad Request [REQUIRED]	`invalid_dpop_proof`	The Credential Issuer cannot fulfill the request because of invalid *DPoP proof*. Section 5 of [**RFC 9449**].
400 Bad Request [REQUIRED]	`use_fresh_attestation`	The Wallet Instance Attestation JWT is not fresh enough to be acceptable by the server. Section 6.2 of OAUTH-ATTESTATION-CLIENT-AUTH.
401 Unauthorized [REQUIRED]	`invalid_client`	The Credential Issuer cannot fulfill the request because of invalid parameters Client Authentication failed (for example in case of unknown client, no parameters Client Authentication included, or unsupported authentication method). (**RFC 6749#section-5.2**).
500 Internal Server Error [REQUIRED]	`server_error`	The Credential Issuer encountered an internal problem.
503 Service Unavailable [REQUIRED]	`temporarily_unavailable`	The Credential Issuer is temporary unavailable.
504 Gateway Timeout [OPTIONAL]	-	The Credential Issuer cannot fulfill the request within the defined time interval.

Access Token

A DPoP-bound Access Token is provided by the Credential Issuer Token endpoint as a result of a successful token request. The Access Token is encoded in JWT format, according to [**RFC 7519**]. The Access Token MUST have at least the following mandatory claims and it MUST be bound to the public key that is provided by the DPoP proof. This binding can be accomplished based on the methodology defined in Section 6 of (**RFC 9449**).

The **DPoP JWT** contains the following JOSE header parameters and claims.

JOSE header	Description	Reference
typ	REQUIRED. It MUST be equal to `at+jwt`.	[**RFC 7515**].

continues on next page

Table 13.34 – continued from previous page

JOSE header	Description	Reference
alg	REQUIRED. A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or with a symmetric algorithm (MAC) identifier.	[**RFC 7515**].
kid	REQUIRED. Unique identifier of the `jwk` used by the Credential Issuer to sign the Access Token.	**RFC 7638#section_3**.

Claim	Description	Reference
iss	REQUIRED. It MUST be an HTTPS URL that uniquely identifies the Credential Issuer. The Wallet Instance MUST verify that this value matches the Credential Issuer where it has requested the credential.	[**RFC 9068**], [**RFC 7519**].
sub	REQUIRED. It identifies the subject of the JWT.	[**RFC 9068**], [**RFC 7519**] and Section 8 of [OIDC].
client_id	REQUIRED. The identifier for the Wallet Instance that requested the Access Token; it MUST be equal to the to kid of the public key of the Wallet Instance specified into the Wallet Instance Attestation (`cnf.jwk`).	[**RFC 9068**], [**RFC 7519**] and Section 8 of [OIDC].
aud	REQUIRED. It MUST be set to the identifier of the Credential Issuer.	[**RFC 9068**].
iat	REQUIRED. UNIX Timestamp with the time of JWT issuance, coded as NumericDate as indicated in **RFC 7519**.	[**RFC 9068**], [**RFC 7519**. Section 4.1.6].
exp	REQUIRED. UNIX Timestamp with the expiry time of the JWT, coded as NumericDate as indicated in **RFC 7519**.	[**RFC 9068**], [**RFC 7519**].
jti	OPTIONAL. It MUST be a String in *uuid4* format. Unique Token ID identifier that the RP SHOULD use to prevent reuse by rejecting the Token ID if already processed.	[**RFC 9068**], [**RFC 7519**].
cnf	REQUIRED. It MUST contain a **jkt** claim being JWK SHA-256 Thumbprint Confirmation Method. The value of the *jkt* member MUST be the base64url encoding (as defined in [**RFC 7515**]) of the JWK SHA-256 Thumbprint of the DPoP public key (in JWK format) to which the Access Token is bound.	[**RFC 9449**. Section 6.1] and [**RFC 7638**].

Refresh Token

A *DPoP-bound Refresh Token* is provided by the Credential Issuer Token endpoint as a result of a successful token request. The Refresh Token is encoded in JWT format, according to [**RFC 7519**]. The Refresh Token MUST have at least the following mandatory claims and it MUST be bound to the public key that is provided by the DPoP proof. This binding can be accomplished based on the methodology defined in Section 6 of (**RFC 9449**).

The **DPoP JWT** MUST contain the following JOSE header parameters and claims.

JOSE header	Description	Reference
typ	It MUST be equal to `rt+jwt`.	[**RFC 7515**].
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or with a symmetric algorithm (MAC) identifier.	[**RFC 7515**].
kid	Unique identifier of the `jwk` used by the Credential Issuer to sign the Access Token.	**RFC 7638#section_3**.

Claim	Description	Reference
iss	It MUST be an HTTPS URL that uniquely identifies the Credential Issuer. The Wallet Instance MUST verify that this value matches the Credential Issuer where it has requested the Credential.	[**RFC 9068**], [**RFC 7519**].
sub	It identifies the subject of the JWT. It MUST be set to the value of the `sub` field in the SD-JWT VC Credential.	[**RFC 9068**], [**RFC 7519**] and Section 8 of [OIDC].
client_id	The identifier for the Wallet Instance that requested the Access Token; it MUST be equal to the to *kid* value identifying the public key used in the Wallet Instance, used in the Wallet Instance Attestation (`cnf.jwk`).	[**RFC 9068**], [**RFC 7519**] and Section 8 of [OIDC].
aud	It MUST be set to the identifier of the Credential Issuer.	[**RFC 9068**].
iat	UNIX Timestamp with the time of JWT issuance, coded as NumericDate as indicated in **RFC 7519**.	[**RFC 9068**], [**RFC 7519**. Section 4.1.6].
nbf	UNIX Timestamp with the time before which the JWT MUST NOT be accepted for processing, coded as NumericDate as indicated in **RFC 7519**. It SHOULD be set to the `exp` claim of the corresponding Access Token.	[**RFC 7519**. Section 4.1.7].

continues on next page

Table 13.37 – continued from previous page

Claim	Description	Reference
exp	UNIX Timestamp with the expiry time of the JWT, coded as NumericDate as indicated in **RFC 7519**.	[**RFC 9068**], [**RFC 7519**].
jti	It MUST be a String in *uuid4* format. Unique Token ID identifier that the RP SHOULD use to prevent reuse by rejecting the Token ID if already processed.	[**RFC 9068**], [**RFC 7519**].
cnf	It MUST contain a **jkt** claim being JWK SHA-256 Thumbprint Confirmation Method. The value of the *jkt* member MUST be the base64url encoding (as defined in [**RFC 7515**]) of the JWK SHA-256 Thumbprint of the DPoP public key (in JWK format) to which the Access Token is bound.	[**RFC 9449**. Section 6.1] and [**RFC 7638**].

Nonce Endpoint

The Nonce Endpoint provides a `c_nonce` value useful to create a proof of possession of key material for the request to the Credential Endpoint, as defined in Section 7 of OpenID4VCI.

Nonce Request

The request for a nonce MUST be an HTTP POST without a body addressed to the Credential Issuer Nonce Endpoint mapped in the Credential Issuer Metadata.

Nonce Response

Nonce Response to the Wallet Instance MUST be sent using *application/json* media type. In case of Nonce Request successful, the Credential Issuer MUST return HTTP response with a *200 (OK)* status code.

As defined in Section 7.2 of OpenID4VCI, the Credential Issuer MUST make the response uncacheable by adding a `Cache-Control` header field valued with *no-store*.

The Nonce Response contains the following parameter:

Claim	Description	Reference
c_nonce	REQUIRED. String containing the nonce value. This value MUST be unpredictable.	Section 7.2 of [OpenID4VCI].

Credential endpoint

The Credential Endpoint issues a Credential upon the presentation of a valid Access Token, as defined in OpenID4VCI.

Credential Request

The Wallet Instance when requests the Digital Credential to the Credential endpoint, MUST use the following parameters in the message body of the HTTP POST request, using the *application/json* media type.

The Credential endpoint MUST accept and validate the *DPoP proof* sent in the DPoP HTTP Header parameter, according to the steps defined in (**RFC 9449**) Section 4.3. The *DPoP proof* in addition to the values that are defined in the Token Endpoint section MUST contain the following claim:

- **ath**: hash value of the Access Token encoded in ASCII. The value MUST use the base64url encoding (as defined in Section 2 of (**RFC 7515**) with the SHA-256 algorithm.

Warning

The Wallet Instance MUST create a **new DPoP proof** for the Credential request and MUST NOT use the previously created proof for the Token Endpoint.

Claim	Description	Reference
credential_identifier	REQUIRED when an Authorization Details of type *openid_credential* was returned from the Token Response. It MUST NOT be used otherwise. This MUST be set with one of the value obtained in the `credential_identifiers` claim of the Token Response. It MUST NOT be used if `credential_configuration_id` is present.	Section 8.2 of [OpenID4VCI].
credential_configuration_id	REQUIRED if `credential_identifiers` parameter is absent in the Token Response. It MUST NOT be used otherwise. String specifying a unique identifier of the Credential being described in the *credential_configurations_supported* map in the Credential Issuer Metadata. For example, in the case of the PID in SD-JWT VC format, it can be set to `dc_sd_jwt_pid`.	Section 8.2 of [OpenID4VCI].

continues on next page

Table 13.38 – continued from previous page

Claim	Description	Reference
proofs	REQUIRED. Object providing one or more proofs of possession of the cryptographic key material to which the issued Credential instances will be bound to. The `proofs` object MUST contain an array parameter named *jwt* containing an array of JWT, where each element within the array is used as proof of possession.	[OpenID4VCI].

The JWT proof type MUST contain the following parameters for the JOSE header and the JWT body:

JOSE Header	Description	Reference
alg	A digital signature algorithm identifier such as per IANA "JSON Web Signature and Encryption Algorithms" registry. It MUST be one of the supported algorithms in Section *Cryptographic Algorithms* (page 615) and MUST NOT be set to `none` or to a symmetric algorithm (MAC) identifier.	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
typ	It MUST be set to *openid4vci-proof+jwt*.	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
jwk	Representing the public key chosen by the Wallet Instance, in JSON Web Key (JWK) [**RFC 7517**] format that the Digital Credential shall be bound to, as defined in Section 4.1.3 of [**RFC 7515**].	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
key_attestation	Representing the Key Attestation.	[OpenID4VCI].

Claim	Description	Reference
iss	The value of this claim MUST be the **client_id** of the Wallet Instance.	[OpenID4VCI], [**RFC 7519**, Section 4.1.1].
aud	It MUST be set to the identifier of the Credential Issuer.	[OpenID4VCI].
iat	UNIX Timestamp with the time of JWT issuance, coded as NumericDate as indicated in **RFC 7519**.	[OpenID4VCI], [**RFC 7519**. Section 4.1.6].
nonce	The value type of this claim MUST be a string, where the value is a **c_nonce** provided by the Credential Issuer in the Nonce Response.	[OpenID4VCI].

Credential Response

Credential Response to the Wallet Instance MUST be sent using *application/json* media type. If the Credential Request is successfully validated, and the Credential is immediately available, the Credential Issuer MUST return HTTP response with a *200 (OK)* status code. If the Credential is not available and the deferred flow is supported by the Credential Issuer, an HTTP status code *202* MUST be returned.

The Credential Response contains the following parameters:

Claim	Description	Reference
credentials	REQUIRED if `interval` and `transaction_id` are not present, otherwise it MUST NOT be present. It is an array of one or more issued Credentials. The number of elements in the Credentials array matches the number of keys that the Wallet Instance has provided either via the `proofs` parameter of the Credential Request. The array MUST contain JSON objects, where each object MUST have exactly one member with the key `credential`. The value of the `credential` member MUST be a string containing the encoded Credential, as further described below: • **credential**: REQUIRED. String containing one issued Digital Credential. If the requested format identifier is `dc+sd-jwt` then the `credential` parameter MUST NOT be re-encoded. If the requested format identifier is `mso_mdoc` then the `credential` parameter MUST be a base64url-encoded representation of the CBOR-encoded IssuerSigned structure, as defined in [ISO 18013-5]. This structure SHOULD contain all Namespaces and IssuerSignedItems that are included in the AuthorizedNamespaces of the MobileSecurityObject.	Section 8.3, Annex A2.4 and Annex A3.4 of [OpenID4VCI].
interval	REQUIRED if `transaction_id` is present, otherwise it MUST NOT be present. The amount of time (in seconds) required before making a Deferred Credential Request (*WP_065* (page 746)).	Section 8.3 of [OpenID4VCI].
notification_id	OPTIONAL. String identifying an issued Credential that the Wallet includes in the Notification Request as defined in Section credential-issuance-endpoint:Notification Request. It MUST NOT be present if the `credentials` parameter is not present.	Section 8.3 of [OpenID4VCI].

continues on next page

Table 13.41 – continued from previous page

Claim	Description	Reference
transaction_id	REQUIRED if `credentials` is not present, otherwise it MUST NOT be present. String identifying a deferred issuance transaction that the Wallet includes in the subsequent Credential Request as defined in Section credential-issuance-endpoint: Deferred Endpoint as per (*WP_065* (page 746)). It MUST be invalidated after the User obtains the Credential.	Section 8.3 of [OpenID4VCI].

In case of the Credential Request does not contain a valid Access Token, the Credential Endpoint returns an error response such as defined in Section 3 of [**RFC 6750**]. If any other error occurs, the Credential Issuer MUST return an error response as defined in Section 8.3.1 of [OpenID4VCI]. The response MUST use the content type *application/json* and MUST include the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of an error response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
    "error": "invalid_proof",
    "error_description": "The proof field is not present or the provided key proof is invalid o
r not bound to a nonce provided by the Credential Issuer."
}
```

In the following table are listed HTTP Status Codes and related error codes that are supported for the error response:

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_credential_request`	The Credential Issuer cannot fulfill the request because of missing parameters, invalid parameters or request malformed. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`unknown_credential_configuration`	The Credential Issuer cannot fulfill the request because the requested Credential Configuration is unknown. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`unknown_credential_identifier`	The Credential Issuer cannot fulfill the request because the requested Credential identifier is unknown. Section 8.3.1 of [OpenID4VCI].

continues on next page

Table 13.42 – continued from previous page

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_proof`	The Credential Issuer cannot fulfill the request because the `proofs` parameter in the Credential Request is invalid or absent or the key proof(s) does not contain the `c_nonce` value. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`invalid_nonce`	The Credential Issuer cannot fulfill the request because the `proofs` parameter in the Credential Request uses an invalid nonce. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`invalid_encryption_parameters`	The Credential Issuer cannot fulfill the request because the encryption parameters in the Credential Request are invalid or missing. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`credential_request_denied`	The Credential Request has not been accepted by the Credential Issuer. The Wallet SHOULD treat this error as unrecoverable, meaning if received from a Credential Issuer the Credential(s) cannot be issued. Section 8.3.1 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`invalid_transaction_id`	Only in case of deferred flow. The Credential Issuer cannot fulfill the request because the Credential Request contains an invalid `transaction_id`. Section 9.3 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`invalid_dpop_proof`	The Credential Issuer cannot fulfill the request because of invalid *DPoP proof*. Section 7 of [**RFC 9449**].
500 Internal Server Error [REQUIRED]	`server_error`	The Credential Issuer encountered an internal problem.
503 Service Unavailable [REQUIRED]	`temporarily_unavailable`	The Credential Issuer is temporary unavailable.
504 Gateway Timeout [OPTIONAL]	-	The Credential Issuer cannot fulfill the request within the defined time interval.

Deferred Endpoint

Credential Issuers MAY support the *Deferred Endpoint* aiming to satisfy the cases where an immediate issuance might be not possible, due to errors during the communication between the Credential Issuer and the Authentic Source (for example the Authentic Source is temporarily unavailable, etc.) or due to administrative or technical processes.

In the case where the Authentic Source and the Credential Issuer are both enabled to use *PDND*, what is described in Section *Authentic Sources* (page 356) MUST apply.

The following requirements apply:

1. The Deferred Credential request MAY also happen several days after the initial Credential request.

2. The User MUST be informed that the Credential is available and ready to be issued.
3. The Wallet Provider MUST NOT be informed about which Credential is available to be issued or which Credential Issuer the User needs to contact.
4. The Wallet Instance MUST be informed about the amount of time to wait before making a new Credential request.
5. As, in general, an unavailability may be an unexpected event, the Credential Issuer MUST be able to switch on the fly between a *immediate* and an *deferred* flow. This decision MUST be taken after the authorization step.

If Credential Issuers, supporting this flow, are not able to immediately issue a requested Credential, they MUST provide the Wallet Instance with an HTTP Credential Response containing the amount of time to wait before making a new Credential request and an identifier of the deferred issuance transaction (*transaction_id*). The HTTP status code MUST be *202* (see Section 15.3.3 of [**RFC 9110**]). Below a non-normative example is given.

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Cache-Control: no-store

{
    "transaction_id": "8xLOxBtZp8",
    "interval": 864000
}
```

The Wallet Instance MUST use the value given in the *interval* parameter to inform the User when the Credential becomes available (e.g. using a local notification triggered by the *interval* time value). Credential Issuers MAY send a notification to the User through a communication channel (e.g. email address), if previously provided by the User to the Credential Issuer.

Deferred Request

Upon receipt of the notification (by the Wallet Instance and/or by the Credential Issuer), the User accesses the Wallet Instance.

The Wallet Instance MUST present to the Deferred Endpoint an Access Token that is valid for the issuance of the Digital Credential previously requested at the Credential Endpoint.

If the `interval` parameter value results as less than the expiration time set for the Access Token, the Wallet Instance SHOULD use the Access Token (*WP_066b* (page 746)). Otherwise, the Wallet Instance MAY obtain a new Access Token following the Refresh Token flow (see Section *Refresh Token Flow* (page 445) for more details as per *WP_066c* (page 746)). If the Refresh Token flow fails, the Wallet Instance needs to submit a new authentication request (*WP_067* (page 746)).

The Deferred Credential Request MUST be an HTTP POST request. It MUST be sent using the `application/json` media type. The following parameter is used in the Deferred Credential Request:

- `transaction_id`: REQUIRED. String identifying a Deferred Issuance transaction (*WP_066a* (page 746)).

The Credential Issuer MUST invalidate the `transaction_id` after the Credential for which it was meant has been obtained by the Wallet Instance. The following is a non-normative example of a Deferred Credential Request:

```
POST /credential HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA


{
  "transaction_id": "8xLOxBtZp8"
}
```

Deferred Response

The Deferred Credential Response MUST be sent using the `application/json` media type.

If the Digital Credential(s) is(are) available, the Deferred Credential Response MUST use the `credentials` and `notification_id` parameters as defined in Section credential-issuance-endpoint: Credential Response and MUST respond with the HTTP status code 200 (see Section 15.3.3 of **RFC 9110**]).

If the Credential Issuer still requires more time, the Deferred Credential Response MUST use the `interval` and `transaction_id` parameters as defined in Section credential-issuance-endpoint: Credential Response and it MUST respond with the HTTP status code 202 (see Section 15.3.3 of **RFC 9110**]).

The value of `transaction_id` MUST be same as the value of `transaction_id` in the Deferred Credential Request.

If the Deferred Credential Request is invalid or the Digital Credential is not available, the Deferred Credential Error Response MUST be sent to the Wallet Instance according to credential-issuance-endpoint:Credential Response.

Notification endpoint

The Notification Endpoint is used by the Wallet to notify the Credential Issuer of certain events for issued Credentials, such as if the Credential was successfully stored in the Wallet Instance.

To uphold privacy, the `event_description` in the notification SHOULD NOT contain any information that could disclose User behavior or reveal the status of the personal device (e.g., storage space full).

This endpoint MUST be protected using a DPoP Access Token. TLS for the confidentiality of the HTTP transport is REQUIRED according to Section 11 of [OpenID4VCI].

Notification Request

The Notification Request MUST be an HTTP POST using the *application/json* media type with the following parameters.

Claim	Description	Reference
notification_id	REQUIRED. It MUST be equal to the `notification_id` value returned in the Credential Response by the Credential Issuer.	Section 11.1 of [OpenID4VCI].
event	REQUIRED. Type of the notification event. It MUST be a case sensitive string and it MUST support the following values: • *credential_accepted*: when the Credential was successfully stored in the Wallet Instance. • *credential_deleted*: when the unsuccessful Credential issuance was caused by a user action. • *credential_failure*: in all other unsuccessful cases.	Section 11.1 of [OpenID4VCI].
event_description	OPTIONAL. Human-readable ASCII [USASCII] text providing additional information, used to inform about the event that occurred. Values for the event_description parameter MUST NOT include characters outside the set *%x20-21 / %x23-5B / %x5D-7E.*	Section 11.1 of [OpenID4VCI].

Notification Response

The Notification Response MUST be use an HTTP status code *204 (No Content)*, as recommended in Section 11.2 of [OpenID4VCI].

In case of errors, what is described in Section 11.3 of [OpenID4VCI] MUST apply.

In case of the Notification Request does not contain a valid Access Token, the Notification Endpoint returns an error response such as defined in Section 3 of [**RFC 6750**]. If any other error occurs, the Credential Issuer MUST return an error response as defined in Section 11.3 of [OpenID4VCI]. The response MUST use the content type *application/json* and MUST include the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of an error response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
    "error": "invalid_notification_id",
    "error_description": "The notification_id field is not valid"
}
```

In the following table are listed HTTP Status Codes and related error codes that are supported for the error response:

Status Code	error code	Description
400 Bad Request [REQUIRED]	`invalid_notification_id`	The Credential Issuer cannot fulfill the request because of invalid `notification_id` parameter. Section 11.3 of [OpenID4VCI].
400 Bad Request [REQUIRED]	`invalid_notification_request`	The Credential Issuer cannot fulfill the request because of missing parameters, invalid parameter or request malformed. Section 11.3 of [OpenID4VCI].
500 Internal Server Error [REQUIRED]	`server_error`	The Credential Issuer encountered an internal problem.

continues on next page

Table 13.44 – continued from previous page

Status Code	error code	Description
503 Service Unavailable [REQUIRED]	`temporarily_unavailable`	The Credential Issuer is temporary unavailable.
504 Gateway Timeout [OPTIONAL]	-	The Credential Issuer cannot fulfill the request within the defined time interval.

Data Correction using credential_failure

According to OpenID4VCI Section 11, in all other unsuccessful cases `event` MUST use `credential_failure` and additional Notification Request parameters MAY be defined and used. The Credential Issuer MUST ignore any unrecognized parameters.

For holder-initiated data correction, the Wallet Instance SHOULD send a Notification Request with `event=credential_deleted` and include additional parameters to signal the data correction context. At minimum:

- `event_description`: concise human-readable description of the discrepancy.
- `failure_reason` (OPTIONAL): a short machine-readable code, e.g., `data_correction_requested`.
- `correction_details` (OPTIONAL): object with minimal fields indicating impacted attributes without sensitive values (e.g., attribute identifiers only).

In the case of receiving a Credential failure that includes a stated reason, the Credential Issuer SHOULD forward the User's message containing that reason to the Authentic Source. It is RECOMMENDED to use PDND specific endpoints, provided by Authentic Source, to facilitate this data exchange.

13.2.3 e-Service PDND Credential Issuer Catalog

Credential Issuers MUST provide the following e-services through PDND to:

- provide statistics about issued Credentials

Get Statistics

Description	This service returns statistical data on issued Digital Credentials.
Provider	Credential Issuer
Consumer	Authorized Third Party

13.3 Relying Party Endpoints

The Relying Party MUST expose a comprehensive set of endpoints to support the complete IT-Wallet ecosystem, including both remote and proximity presentation flows.

13.3.1 Relying Party Remote Flow Endpoints

The Relying Party MUST expose a set of endpoints to support remote presentation flows as defined in OpenID4VP 1.0. These endpoints enable secure credential verification, trust establishment, and user authentication for cross-device and same-device interaction patterns.

Note

Tests related to Relying Party remote flow endpoints are defined in the remote presentation test matrix (test-plans-remote-presentation:Remote Credential Verifier Test Matrix).

Federation Endpoint

The Relying Party MUST provide its Entity Configuration through the `/.well-known/openid-federation` endpoint, according to Section *Entity Configuration* (page 117). This endpoint enables trust establishment and discovery of the Relying Party's capabilities.

Technical details are provided in Section *Relying Party Entity Configuration* (page 352).

OpenID4VP Remote Flow Endpoints

The following endpoints are required for OpenID4VP 1.0 remote presentation flows as described in *Remote Flow* (page 483). These endpoints support both Same Device and Cross Device flows:

Request URI Endpoint

The Request URI Endpoint is where the Relying Party provides the signed Request Object to the Wallet Instance. This endpoint supports both GET and POST methods as defined in the OpenID4VP 1.0 specification.

For detailed implementation requirements, see *Request URI Request* (page 495) and *Request URI Response* (page 497).

Response URI Endpoint

The Response URI Endpoint receives the Authorization Response from the Wallet Instance containing the Verifiable Presentation. This endpoint processes the presentation and validates the credentials.

For detailed implementation requirements, see *Authorization Response* (page 502) and *Relying Party Response* (page 506).

Status Endpoint (Optional)

The Status Endpoint is an optional endpoint that allows the user-agent to monitor the progress of the presentation flow. This endpoint is particularly useful for Same Device flows where the user-agent needs to know when the Wallet Instance has completed the presentation.

For detailed implementation requirements, see *Status Endpoint* (page 508) and *Status Endpoint Errors* (page 509).

User Data Management Endpoints

The following endpoint supports user data management and privacy compliance requirements for remote flows:

Relying Party Erasure Endpoint

The Erasure Endpoint, which is described in *Relying Party Metadata* (page 354), allows Wallet Instances to request deletion of attributes presented to the Relying Party. The Relying Party MUST request User authentication before proceeding with the attribute deletion.

Erasure Request

The Erasure Request MUST be a GET request to the Erasure Endpoint. The Wallet Instance MUST also support a call back mechanism which enables the User-Agent to notify the Wallet Instance (and thus the User) once the Erasure Response is returned.

Below is a non-normative example of an Erasure Request where the call back URL is passed as a query parameter.

```
GET /erasure-endpoint?callback_url=https://wallet-instance/erasure_response HTTP/1.1
Host: relying-party.example.org
```

Erasure Response

If the deletion of all attributes bound to the User have been successful, the Erasure Response MUST return a 204 HTTP status code.

If instead the attributes deletion procedure fails due any circumstances, the Relying Party MUST return an error response with `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters (e.g., header parameters or integrity assertion), or includes invalid and unknown parameters.
`401 Unauthorized`	`unauthorized`	The request could not be fulfilled due to invalid authentication by the User.
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Erasure Endpoint encountered an internal problem. (**RFC 6749#section-4.1.2.1**).
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Erasure Endpoint is temporarily unavailable (e.g., due to maintenance or overload). (**RFC 6749#section-4.1.2.1**).

The following is an example of an error response from Erasure Endpoint:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
```

```
{
 "error": "server_error",
 "error_description": "The request cannot be fulfilled due to an internal server error."
}
```

Upon receiving an error response, the Wallet Instance which made the Erasure Request MUST inform the User of the error condition in an appropriate manner.

Security Considerations

All Relying Party endpoints MUST implement appropriate security measures:

- **HTTPS Only**: All endpoints MUST be accessible only over HTTPS
- **Endpoint Mix-up Protection**: Endpoint URLs MUST be attested by trusted third parties through the Trust Chain
- **Input Validation**: All endpoints MUST validate input parameters and reject malformed requests
- **Rate Limiting**: Endpoints SHOULD implement rate limiting to prevent abuse
- **Audit Logging**: All endpoint interactions SHOULD be logged for security monitoring

For detailed security requirements, see *Remote Flow* (page 483) and the relevant test cases in test-plans-remote-presentation:Remote Credential Verifier Test Matrix.

Implementation Notes

- The specific implementation details for most endpoints are left to the Relying Party's discretion
- Endpoints MUST comply with the OpenID4VP 1.0 specification for remote flows
- Proximity flow endpoints MUST support the lifecycle management of Verifier Apps
- All endpoints MUST be discoverable through the Relying Party's Entity Configuration
- Error responses MUST follow the standard HTTP status codes and include appropriate error descriptions

For comprehensive implementation guidance, refer to the individual endpoint sections and the test matrices for validation requirements.

13.3.2 Relying Party Provider Backend Endpoints

The Relying Party MUST expose a set of endpoints for handling the lifecycle of Verifier Apps that use a remote backend service provided by their Relying Party Provider Backend. These endpoints support proximity presentation flows by providing nonce generation, hardware key registration, integrity validation, and Access Certificate issuance. The specific implementation details are left to the Relying Party's discretion.

Note

Tests related to Relying Party endpoints are defined in the remote presentation test matrix (test-plans-remote-presentation:Remote Credential Verifier Test Matrix) and proximity presentation test matrix (test-plans-proximity-presentation:Proximity Credential Verifier Test Matrix).

Relying Party Provider Backend Federation Endpoint

The Relying Party MUST provide its Entity Configuration through the `/.well-known/openid-federation` endpoint, according to Section *Entity Configuration* (page 117). Technical details are provided in Section *Relying Party Entity Configuration* (page 352).

Relying Party Provider Backend Nonce Endpoint

The Relying Party Nonce Endpoint allows the Verifier App to request a cryptographic `nonce` from the Relying Party Provider Backend. The `nonce` serves as an unpredictable, single-use challenge to ensure freshness and prevent replay attacks.

Further details on the Nonce Request and Response are provided in the *Mobile Application Nonce Request* (page 672) and *Mobile Application Nonce Request* (page 672) Sections, respectively.

Relying Party Verifier App Initialization Endpoint

The Verifier App Initialization Endpoint allows for the initialization of Verifier Apps, consisting in the registration of a pair of long-lived, securely stored Cryptographic Hardware Keys.

Further details on the Verifier App Initialization Request and Response are provided in the *Mobile Application Instance Initialization Request* (page 673) and *Mobile Application Instance Initialization Response* (page 674) Sections, respectively.

Relying Party Provider Backend Key Binding Endpoint

The Relying Party Key Binding Endpoint enables Verifier Apps to bind the newly created pair of keys, which will be associated with an Access Certificate, to the Verifier App, by relying on a proof of possession of the Cryptographic Hardware Keys generated during the *Mobile Application Instance Initialization* (page 667) phase. Before completing the process, the Relying Party Provider Backend also needs to verify the integrity of the Verifier App.

Relying Party Provider Backend Key Binding Request

Further details on the Relying Party Key Binding Request are provided in the *Wallet Instance Attestation Issuance Request* (page 547) section.

The only difference are the following:

- The `typ` header of the Integrity Request JWT assumes the value `rp-kb+jwt`, and
- The `hardware_signature` claim value is obtained based on only `client_data_hash` value.

Relying Party Provider Backend Key Binding Response

Upon a successful request, the Relying Party Provider Backend provides an HTTP Response with a `204 No Content` status code.

Below is a non-normative example of a Key Binding Request Response.

```
HTTP/1.1 204 No content
```

If any errors occur during the process, an error response is returned. The response uses `application/json` as the `Content-Type` and includes the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered (*WP_035* (page 739)).

Below is a non-normative example of a Key Binding Error Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The provided challenge is invalid, expired, or already used."
}
```

The following table lists HTTP Status Codes and related error codes that are supported for the error response, unless otherwise specified (*WP_036–0339* (page 739) and *WP_150–155* (page 767)):

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters (e.g., header parameters or Integrity Assertion), or includes invalid and unknown parameters.
`403 Forbidden`	`invalid_request`	The Verifier App has been revoked.
`403 Forbidden`	`integrity_check_error`	The device does not meet the Relying Party Provider's minimum security requirements.
`403 Forbidden`	`invalid_request`	The signature of the Integrity Request is invalid or does not match the associated public key (JWK).
`403 Forbidden`	`invalid_request`	The Integrity Assertion validation failed; the Integrity Assertion is tampered with or improperly signed.
`403 Forbidden`	`invalid_request`	The provided `nonce` is invalid, expired, or already used.
`403 Forbidden`	`invalid_request`	The Proof of Possession (`hardware_signature`) is invalid.
`403 Forbidden`	`invalid_request`	The `iss` parameter does not match the Relying Party Provider's expected URL identifier.
`404 Not Found`	`not_found`	The Verifier App Instance was not found.
`422 Unprocessable Content` [OPTIONAL]	`validation_error`	The request does not adhere to the required format.
`500 Internal Server Error`	`server_error`	An internal server error occurred while processing the request.
`503 Service Unavailable`	`temporarily_unavailable`	The service is unavailable. Please try again later.

Relying Party Provider Backend Access Certificate Endpoint

The Relying Party Access Certificate Endpoint enables Verifier Apps to obtain an Access Certificate.

Relying Party Provider Backend Access Certificate Request

The Access Certificate Request uses the HTTP POST method with `Content-Type` set to `application/json`.

The request includes the following body parameter:

Parameter	Description	Reference
csr	The CSR generated by the Verifier App, encoded in the `base64url` format as defined in **RFC 2511**.	

Below is a non-normative example of an Access Certificate Request.

```
POST /access-certificate HTTP/1.1
Host: relying-party.example.org
Content-Type: application/json

{
  "csr": "MIIBvzCCAa..."
}
```

Relying Party Provider Backend Access Certificate Response

Upon a successful request, the Relying Party Access Certificate Endpoint provides an HTTP Response with a `200 OK` status code and the Access Certificate. The Access Certificate Response, which uses `application/json` as the `Content-Type`, includes the following body parameters:

Parameter	Description	Reference
access_certificate	The Access Certificate generated by the CSR.	This specification.

Below is a non-normative example of an Access Certificate Response.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_certificate": "hajdnhaghSDGns..."
}
```

If any errors occur, the Relying Party Access Certificate Endpoint returns an error response. The response uses `application/json` as the content type and includes the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of an Access Certificate Error Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
```

```
{
    "error": "invalid_request",
    "error_description": "The public key in the CSR is different from the one associated with t
he Cryptographic Hardware Keys."
}
```

The following table lists HTTP Status Codes and related error codes that MUST be supported for the error response, unless otherwise specified:

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters (e.g., header parameters or integrity assertion), or includes invalid and unknown parameters.
`403 Forbidden`	`invalid_request`	The public key in the CSR does not match the public key associated with the Cryptographic Hardware Keys.
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Endpoint encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Endpoint is temporarily unavailable (e.g., due to maintenance or overload).

Relying Party Provider Backend Erasure Endpoint

The Erasure Endpoint allows Wallet Instances to request deletion of attributes presented to the Relying Party, supporting user privacy rights and regulatory compliance.

For detailed implementation requirements, see relying-party-remote-flow-endpoints:Relying Party Erasure Endpoint.

13.4 Authentic Source Endpoints

13.4.1 e-Service PDND Authentic Source Catalog

Public Authentic Sources MUST provide the following e-Service through PDND to provide the Credential Issuer with User's attributes required to the issuance of a Digital Credential.

The e-service is described via an OpenAPI definition where the request, response, and error messages are detailed.

Note

The base OpenAPI Specification is available *Authentic Source PDND OpenAPI Specification* (page 713). This OpenAPI specification can be extended by the Authentic Sources, in fact, the array `attributeClaims` MAY contain additional properties specific to a particular Credential. These additional properties, as defined in the OpenAPI specification, will be inserted into the Credential by the Credential Issuer.

Get Attribute Claims

Description	This e-Service provides the Credential Issuer with all attribute claims necessary for the issuance of a Digital Credential.
Provider	Authentic Source
Consumer	Credential Issuer

This e-Service MUST be invoked by the Credential Issuer in the following cases:

- During the Credential issuance flow to retrieve the Digital Credential dataset(s). In this situation, the `object_id` claim MUST NOT be used in the e-service request payload, and the e-service response MUST include all Credential datasets that are `VALID` state and not administratively expired (i.e., `expiry_date` > current date of the e-service request).
- After receiving an update notification through Signal Hub. In this situation, the `object_id` claim MUST be included in the request payload, and the e-service response MUST return only the Credential dataset identified by the `object_id` claim independently from its state.

Note

The Authentic Source and the Credential Issuer MUST implement the necessary logic to keep track of the requests and responses exchanged via this e-Service, in order to be able to correlate them with the related issuance of a Digital Credential. In particular,

- both MUST save the `object_id` value in the payload of the response to manage Signals related to the deferred issuance of a Digital Credential (see *Signals Processing* (page 613)). Any subsequent notification related to a specific dataset MUST be handled via `object_id`.
- the Authentic Source MUST record the datetime value provided within the `last_updated` parameter, which indicates the last time the User's attributes were updated in the Authentic Source's database;

- the Credential Issuer MUST read the `last_updated` value received in the response to be able to check if the User's attributes have changed since the last issuance of a Digital Credential.

Credential Lifecycle States Mapping

To ensure consistency between the "Electronic Attestation Lifecycle" documented in *Digital Credential Lifecycle* (page 398) and the OpenAPI status Enum, the following mapping and operational logic MUST be applied for the `status` field in the `attributeClaims`.

Directionality and Responsibility: State changes of a Digital Credential at the Credential Issuer level DO NOT imply a change at the Authentic Source. Conversely, any state change of a dataset at the Authentic Source MUST be processed by the Credential Issuer to update the technical Digital Credential status.

Operational Guidance:

- **Technical vs Administrative Validity**: Digital Credentials distinguish between a **technical validity** set by the Credential Issuer (claims `iat` and `exp`) and an **administrative validity** determined by the Authentic Source (claims `issuance_date` and `expiry_date`).
- **Expiry Hierarchy**: The technical expiry (`exp`) MUST NOT be later than the administrative expiry (`expiry_date`). For example, a driver's license may be administratively valid for 10 years, while the issued Digital Credential may have a technical expiry of 1 year.
- **Re-issuance**: If a Digital Credential reaches its technical expiry (`exp`) but the dataset is still administratively valid, the OpenAPI status remains `VALID`, allowing the Digital Credential to be re-issued multiple times within the administrative timeframe.
- **Metadata Verification**: The Credential Issuer MUST verify the effective usability by checking both the technical claims and the administrative dates.
- **Irreversibility**: After a transition to `INVALID`, the Digital Credential cannot return to a `VALID` state. A new issuance is required for a new dataset. This applies to both explicit revocation and administrative expiry.
- **Signal Processing**: Signals MUST be processed sequentially. If a Signal invalidates a Digital Credential, subsequent correction Signals for the same object are ignored.

Status Mapping and Case Logic:

The Credential Issuer MUST update the Digital Credential `status` based on Signals received from the Authentic Source via Signal Hub (`signalType=UPDATE`):

- **Revocation**: If a dataset is revoked at the Authentic Source (status `INVALID`), the Credential Issuer MUST revoke the Digital Credentials that use that dataset (status transition from `VALID/SUSPENDED` to `INVALID`).
- **Suspension**: If a dataset is suspended at the Authentic Source (status `SUSPENDED`), the Credential Issuer MUST suspend the Digital Credentials (status transition from `VALID` to `SUSPENDED`).
- **Restoration**: If a suspended dataset returns to `VALID` at the Authentic Source, the Credential Issuer MUST restore the Digital Credential validity (status transition from `SUSPENDED` to `VALID`).
- **Modification**: If a dataset is modified but remains `VALID` at the Authentic Source, the Credential Issuer—detecting the change via the `last_updated` field—assigns the technical status `ATTRIBUTE_UPDATE` to the Digital Credential. This triggers a re-issuance flow when the Wallet Instance checks the status.
- **Administrative Expiry:**
  - **Scenario A (Metadata-driven)**: If `expiry_date` was shared with the Credential Issuer in metadata, the Authentic Source DOES NOT send Signals upon expiry; the Credential Issuer manages the lifecycle independently ensuring technical `exp` <= `expiry_date`.
  - **Scenario B (Signal-driven)**: If `expiry_date` is NOT present in metadata and the dataset expires, the Authentic Source MUST set its status to `INVALID` and send a Signal via Signal Hub; the Credential Issuer then revokes the Digital Credentials (status transition to `INVALID`).

Example of Authentic Source response

The endpoint response MUST use the HTTP Content-Type set to `application/json`. Below is a concrete example with fictitious data to clarify the expected shape and content.


```
{
  "userClaims": {
    "given_name": "Mario",
    "family_name": "Rossi",
    "birth_date": "1980-01-10",
    "birth_place": "Roma",
    "tax_id_code": "TINIT-RSSMRA80A01H501Z",
    "personal_administrative_number": "12345A123A"
  },
  "attributeClaims": [
    {
      "object_id": "6F9619FF-8B86-D011-B42D-00C04FC964FF",
      "institute_name": "Nome Istituto Universitario",
```

```
      "programme_type_name": "Laurea Magistrale",
      "degree_course_name": "Computer Science - Informatica",
      "academic_qualification_date": "2025-06-25",
      "issuance_date": "2025-06-25",
      "...": ""
    },
    {
      "object_id": "7A0720AB-9C97-E122-C53E-11D05FD075GG",
      "institute_name": "Nome Istituto Universitario",
      "programme_type_name": "Laurea Triennale",
      "degree_course_name": "Informatica",
      "academic_qualification_date": "2022-11-27",
      "issuance_date": "2022-11-27",
      "...": ""
    }
  ],
  "metadataClaims": [
    {
      "object_id": "6F9619FF-8B86-D011-B42D-00C04FC964FF",
      "description": "Laurea Magistrale in Informatica 2025",
      "status": "VALID",
      "last_updated": "2025-09-15T10:30:00Z"
    },
    {
      "object_id": "7A0720AB-9C97-E122-C53E-11D05FD075GG",
      "description": "Laurea Triennale in Informatica 2027",
      "status": "VALID",
      "last_updated": "2023-01-10T08:00:00Z"
    }
  ]
}
```


In summary:

- **userClaims**: user identity data (first name, family name, date/place of birth, tax id or personal administrative number). At least one of `tax_id_code` and `personal_administrative_number` is required when providing user claims.
- **attributeClaims**: array of datasets; each element **MUST** contain `object_id`, `status` (VALID | INVALID | SUSPENDED), `last_updated` (ISO 8601 format), plus any dataset-specific attributes (e.g. `nationality`, `residence_address`).
- **metadataClaims**: array of metadata per dataset (`object_id` required; `issuance_date` and `expiry_date` optional).
- **interval**: required when the request does not include a `claims` parameter; indicates the number of seconds to wait before repeating the request (e.g. 864000 = 10 days).

The successful response (HTTP 200) returns a `CredentialClaimsResponse` object formatted as a **JSON Payload**, accompanied by the `Agid-JWT-Signature` and `Digest` integrity headers.

Signature Verification and Key Management

The Credential Issuer (Consumer) MUST verify the integrity and authenticity of the JSON response by validating the `Agid-JWT-Signature` and `Digest` headers received from the Authentic Source.

The signature verification and key retrieval process MUST strictly follow the **INTEGRITY_REST_02** standard pattern defined for **PDND e-Services**. Please refer to the Technical Appendix (Section *e-Service PDND* (page 675)) for details on JWT signature validation and specifications for retrieving the Provider's public key via the Interoperability API.

Warning

Alternative mechanisms for distributing cryptographic material (e.g., public `.well-known` endpoints directly exposed by the Authentic Source or *out-of-band* distribution) are not allowed. Trust management MUST remain centralized within the perimeter of the PDND infrastructure as described in the references cited above.

13.5 PDND Signal Hub Endpoints

Within the PDND platform, the Signal Hub serves as an intermediary component between PDND Providers and their PDND Consumers to facilitate real-time data variation notifications. To enable this functionality, the PDND Manager, acting in the role of e-Service PDNDProvider, offers the following specialized PDND Signal Hub e-Services:

- the Signal Collection e-Service which is used by e-Service PDNDProviders to deposit Signals, since the e-Service PDNDProvider acts as Consumer of the Signal Collection e-Service;
- the Signal Distribution e-Service which is used by e-Service PDNDConsumers to retrieve collected Signals, since the e-Service PDNDConsumer also acts as Consumer of the Signal Distribution e-Service.

In order to protect the privacy of the Signal's subject, the PDND Manager requires each e-Service PDNDProvider to pseudonymize the subject's identifier used within the Signals and set up a pseudonymization endpoint for their PDND e-Service. This pseudonymization endpoint is used by e-Service Consumers to use the pseudonymization algorithm in order to calculate the Signal subject's pseudonym. Only the e-Service PDNDProvider and its e-Service PDNDConsumers

are able to link a Signal to the subject's personal data, while the PDND Manager only handles pseudonymized identifiers.

For detailed technical specifications and implementation guidelines, please refer to the Signal Hub Guide.

In the context of the IT Wallet, Authentic Sources interact with the Signal Hub to notify Credential Issuers about changes in the status and/or value of attributes associated with Digital Credentials. Specifically,

- the Authentic Source will deposit Signals in the Signal Hub, thus playing the role of PDND Consumer of the Signal Collection e-Service;
- the Credential Issuer will retrieve Signals from the Signal Hub, and will thus play the role of e-Service PDNDConsumer of the Signal Distribution e-Service.

Note

In the context of IT Wallet, due to the particular nature of the data exchanged, the pseudonymization of the Signal's subject is not needed since it is already an opaque identifier unrelated to the Digital Credential's subject. Thus, the Authentic Source does not need to set up a pseudonymization endpoint for its e-Services.

Authentic Sources that leverage PDND MUST use the Signal Hub e-Services.

Below it is provided the description of how the Authentic Sources and Credential Issuers interact with the Signal Hub e-Services together with the detailed formats of the requests and responses.

13.5.1 Prerequisites

Before using the Signal Hub, all Authentic Sources MUST:

- have registered as Providers of their e-Service with the PDND;
- have registered as Consumers of the Signal Collection e-Service of the Signal Hub;

Before using the Signal Hub, all Credential Issuers MUST:

- have registered as Consumers of the relevant Authentic Sources' e-Services;
- have registered as Consumers of the Signal Distribution e-Service of the Signal Hub;

13.5.2 Signal Hub e-Services

This section describes the endpoints available for the Signal Hub Collection and Distribution e-Services, including their functionalities and the expected request and response formats.

Signal Collection e-Service

Authentic Sources in the IT Wallet ecosystem use the Signal Collection e-Service to:

- notify the Credential Issuer of a change of status and/or value of a specific attribute associated with a Digital Credential issued by the Credential Issuer;
- notify the Credential Issuer of the availability of the attributes related to a specific Digital Credential which a User requested in its Wallet.

The last case, referred to as deferred issuance, happens when the Credential Issuer has requested a Digital Credential's attributes from the Authentic Source (invoking the *Get Attribute Claims* (page 603) PDND endpoint) and the Authentic Source cannot respond immediately with the requested attributes. Thus, the Authentic Source notifies the Credential Issuer via the Signal Hub at a later time that the requested attributes are now available.

The Signal Collection e-Service endpoint is used by Authentic Sources to deposit Signals to the Signal Hub via a Signal Collection request. The latter MUST be a POST request with `Content-Type` set to `application/json`, whose header MUST have the parameters described in Signal Hub Guide and whose body MUST contain the following parameters:

Parameter Name	Description
signalId	REQUIRED. A positive 64 bit integer number referencing the identifier of the Signal in chronological order.
objectType	REQUIRED. Using this field the Authentic Source MAY use to further specify the Signal.
objectId	REQUIRED. The subject to which the Signal is bound. It MUST be set to the Authentic Source's *Object_id* (`object_id`) used in the *get attributes* payload response to the Credential Issuer (see *Get Attribute Claims* (page 603)). The Signal `signalType` MUST be valued with one of the following: • `CREATE`, in order to notify the availability of the attributes related to a specific Digital Credential. • `UPDATE`, in order to notify the updating of the attributes related to a specific Digital Credential.
signalType	REQUIRED. Signal Type. It MUST be one of the following: • `UPDATE`, when the Signal refers to a change in the status and/or value of a specific attribute associated with a Digital Credential; • `CREATE`, when the Signal refers to the availability of the attributes of a specific Digital Credential;
eserviceId	REQUIRED. e-Service to which the Signal is bound. It MUST correspond to the e-Service Id value the Authentic Source is a Provider of.

Note

In the deferred issuance flow, i.e., when the Authentic Source notifies the Credential Issuer of the availability of the Digital Credential's attributes via Signal Hub; both entities MUST keep track of the Authentic Source's `object_id` value used in the `get attributes` payload response. This is necessary since the `objectId` of the Signal MUST be set to that `object_id` value when the Signal has `signalType` with value `CREATE`.

A non-normative example of the Signal Collection request can be found at Signal Hub push.

The Signal Collection e-Service response, acknowledging the correct parsing of the request, is specified in Signal Hub push, and has `Content-Type` set to `application/json`. The payload contains the body parameter:

Parameter Name	Description
signalId	REQUIRED. The identifier of the Signal that has been successfully collected by the Signal Collection e-Service.

If any error occurs during the request parsing, the response MUST adhere to the error format defined in Signal Hub push-yaml.

Note

A complete OpenAPI Specification of the Signal Collection e-Service is available Signal Hub push-yaml.

The Authentic Source MUST implement the necessary logic to handle the requests to the Signal Collection e-Service endpoint, in doing this it has to consider the following aspects:

- Signals are sent per PDND e-Service, therefore the Authentic Source SHOULD implement a Signal deposit cycle for each e-Service ID it is a PDND Provider of;
- Signals are labeled by a unique identifier, the `signalId`, which is a positive 64 bit integer number. The `signalId` MUST be incremented by 1 for each new Signal the Authentic Source wishes to deposit in the Signal Collection e-Service endpoint. It is up to the Authentic Source to keep track of the last `signalId` it has sent. Signals with lower `signalId` values are considered older by the Signal Collection e-Service endpoint and raise an error when received.

Signal Distribution e-Service

Credential Issuers in the IT Wallet ecosystem use the Signal Distribution e-Service to:

- check for changes in the status and/or value of a specific attribute associated with a Digital Credential issued by the Credential Issuer itself;
- check for the availability of attributes related to a Digital Credential requested by a User.

The Signal Distribution e-Service endpoint is used by Credential Issuers to retrieve Signals from the Signal Hub via a Signal Distribution request. The latter MUST use an HTTP request using the method GET and the following parameters:

- Path Parameters:
  - `eserviceId`. REQUIRED. e-Service to which the Signal is bound. It MUST correspond to the e-Service Id value the Credential Issuer is a Consumer of.
- Query Parameters:
  - `signalId`. OPTIONAL. Integer representing the last Signal number processed by the Credential Issuer. The Signal Distribution e-Service will respond with Signals having progressively greater `signalId` values. If not specified, the default value is the lowest `signalId` value available in the Signal Distribution e-Service.

  - `size`. OPTIONAL. Integer representing the maximum number of Signals to be returned in the Signal Distribution response. If not specified, the default value is `10`.

- Headers parameters: these are those described in Signal Hub pull.

If the Signal Distribution request is correctly processed, the e-Service will then respond with status code

- HTTP 200 OK if the request is correctly formed and there are no more Signals to request;
- HTTP 206 OK if the request is correctly formed but there are more Signals to request.

Regardless of the response code used, the response has `Content-Type` set to `application/json` an the header parameters indicated in Signal Hub pull. The body parameter `lastSignalId`, referencing the `signalId` of the last Signal transmitted by the Signal Distribution e-Service is added to the response's payload.

In Signal Hub pull can be found non-normative examples of a Signal Distribution requests and responses.

If any error occurs during the request parsing, the response MUST adhere to the error format defined in Signal Hub pull-yaml.

Note

A complete OpenAPI Specification of the Signal Collection e-Service is available Signal Hub pull-yaml.

The Credential Issuer MUST implement the necessary logic to handle the Polling of the Signal Distribution e-Service endpoint, in doing this it has to consider the following aspects:

- Signals are queried and retrieved per PDND e-Service, meaning that the Credential Issuer MUST implement a poll cycle for each e-Service ID;
- the retention period of Signals in the Signal Hub is specified in Signal Hub Guide. Signals older than the specified retention period are not available for retrieval;
- the Signal Hub does not keep track of the last `signalId` notified to a specific Credential Issuer. Each Credential Issuer MUST keep track of the last `signalId` it has received for each e-Service PDNDID;
- the Signal Distribution e-Service endpoint returns Signals in batches. The maximal size the batch is specified in Signal Hub Guide. The Signal Distribution e-Service will indicate if additional Signals are available for retrieval.

13.5.3 Signals Processing

After the Signals have been successfully recovered by the Credential Issuer, the latter MUST process them as follows:

- For each Signal, the Credential Issuer MUST check the `SignalType` value:
  - if the Signal `SignalType` is `UPDATE` (where `objectId` refers to the *Object_id* of the Authentic Source), the status and/or value of the attribute associated with a Digital Credential need updates;
  - if the Signal `SignalType` is `CREATE` (where `objectId` refers to the *Object_id* of the Authentic Source), the requested attributes of a specific Digital Credential are now available;

  If the `objectId` does not correspond to any valid identifier known to the Credential Issuer, the Signal MUST be ignored. Otherwise, if it corresponds to a known and valid identifier, the Credential Issuer MUST use the *Get Attribute Claims* (page 603) PDND endpoint of the Authentic Source to retrieve the updated information and, if applicable, apply the new status to the corresponding Credential.

  When the Signal has been processed, the Credential Issuer will either move to the next Signal and update its `signalId` counter; or, if there are no more Signals to process, it will resume the Pull cycle.

Warning

Given Signal Hub's currently supported security patterns, if the Authentic Source requires the *AUDIT_REST_02* security pattern from the Credential Issuer, the latter MUST revoke the Digital Credential referenced in Signals with `signalType UPDATE` since it cannot contact the Authentic Source to retrieve the updated information without having authenticated the User before. **In this scenario, revocation is the only allowed action.**

CHAPTER

FOURTEEN

CRYPTOGRAPHIC ALGORITHMS

This section lists cryptographic algorithms used in the IT Wallet ecosystem. Algorithm identifiers are primarily expressed as JOSE `alg` values; when COSE is used, the corresponding COSE algorithms are explicitly marked (e.g., ESP256 vs ES256).

The following algorithms MUST be supported:

Algorithm 'alg' parameter value	Description	Operations	References
ES256	Elliptic Curve Digital Signature Algorithm (ECDSA) using P-256 and SHA-256.	Signature	RFC 7518.
ES384	ECDSA using P-384 and SHA-384.	Signature	RFC 7518.
ES512	ECDSA using P-521 and SHA-512.	Signature	RFC 7518.
ESP256 (COSE, `-9`)	ECDSA using P-256 and SHA-256.	Signature	RFC 9864.
ESP384 (COSE, `-51`)	ECDSA using P-384 and SHA-384.	Signature	RFC 9864.
ESP512 (COSE, `-52`)	ECDSA using P-521 and SHA-512.	Signature	RFC 9864.
RSA-OAEP-256	RSA Encryption Scheme with Optimal Asymmetric Encryption Padding (OAEP) using SHA-256 hash function and the MGF1 with SHA-256 mask generation function.	Key Encryption	RFC 7516, RFC 7518.

continues on next page

Table 14.1 – continued from previous page

Algorithm 'alg' parameter value	Description	Operations	References
ECDH-ES	Elliptic Curve Diffie-Hellman (ECDH) Ephemeral-Static key agreement.	Key Agreement	**RFC 7516**, **RFC 7518**.
A128CBC-HS256	AES encryption in Cipher Block Chaining mode with 128-bit Initial Vector value, plus HMAC authentication using SHA-256 and truncating HMAC to 128 bits.	Content Encryption	**RFC 7516**, **RFC 7518**.
A256CBC-HS512	AES encryption in Cipher Block Chaining mode with 256-bit Initial Vector value, plus HMAC authentication using SHA-512 and truncating HMAC to 256 bits.	Content Encryption	**RFC 7516**, **RFC 7518**.
A128GCM	AES encryption with Galois/Counter Mode and a key length of 128.	Content Encryption	**RFC 7516**, **RFC 7518**.
A256GCM	AES encryption with Galois/Counter Mode and a key length of 256.	Content Encryption	**RFC 7516**, **RFC 7518**.

For Credentials issued in mdoc format, the following algorithms MUST be supported:

Algorithm	Description	Operations	References
ECKA-DH	Elliptic Curve Key Agreement Algorithm – Diffie-Hellman.	Key agreement	BSI TR-03111.
HKDF	HMAC-based Key Derivation Function.	Session key derivation	**RFC 5869**.
AES-256-GCM	Advanced Encryption Standard with Galois/Counter Mode and a key length of 256.	Session encryption	NIST SP 800-38D.

The following algorithms are RECOMMENDED to be supported:

Algorithm 'alg' parameter value	Description	Operations	References
PS256	RSASSA (RSA with Signature Scheme Appendix) with PSS (Probabilistic Signature Scheme) padding using SHA-256 hash function and MGF1 mask generation function with SHA-256.	Signature	**RFC 7518**.
PS384	RSASSA with PSS padding using SHA-384 hash function and MGF1 mask generation function with SHA-384.	Signature	**RFC 7518**.
PS512	RSASSA with PSS padding using SHA-512 hash function and MGF1 mask generation function with SHA-512.	Signature	**RFC 7518**.
ECDH-ES	Elliptic Curve Diffie-Hellman (ECDH) Ephemeral Static key agreement using Concat Key Derivation Function (KDF).	Key Encryption	**RFC 7518**.
ECDH-ES+ A128KW	ECDH-ES using Concat KDF and content encryption key (CEK) wrapped using AES with a key length of 128 (A128KW).	Key Encryption	**RFC 7518**.
ECDH-ES+ A256KW	ECDH-ES using Concat KDF and content encryption key (CEK) wrapped using AES with a key length of 256 (A256KW).	Key Encryption	**RFC 7518**.

The following algorithms MUST NOT be supported:

Algorithm 'alg' parameter value	Description	Operations	References
none	•	Signature	**RFC 7518**.
RSA_1_5	RSAES with PKCS1-v1_5 padding scheme. Use of this algorithm is generally not recommended.	Key Encryption	**RFC 7516**, [Security Vulnerability], [SOG-IS].
RSA-OAEP	RSA Encryption Scheme with Optimal Asymmetric Encryption Padding (OAEP) using default parameters.	Key Encryption	**RFC 7518**, [SOG-IS].
HS256	HMAC using SHA-256.	Signature	**RFC 7518**.
HS384	HMAC using SHA-384.	Signature	**RFC 7518**.
HS512	HMAC using SHA-512.	Signature	**RFC 7518**.

CHAPTER

FIFTEEN

SECURITY AND PRIVACY CONSIDERATIONS

This section provides an informal security analysis of the IT-Wallet specification by analyzing the compliance with the security and privacy requirements identified in [OpenID4VC-SecTrust].

Note

As [OpenID4VC-SecTrust] is still a work in progress, the security and privacy considerations described may change in the future.

Note

The focus of the analysis is the compliance of the design choices in the IT-Wallet specification with respect to the OpenID4VC protocols. It is currently out-of-scope *(i)* the analysis of the design of the proximity flow based on ISO 18013-5, and *(ii)* the analysis of the implementation; as a consequence 7 requirements specifically related to the implementation are not considered (e.g., SV-00: The Verifier must implement the protocol securely and correctly).

As in [OpenID4VC-SecTrust], all requirements are numbered for reference. Together with the respective component that needs to implement the requirement:

- **CF**: Credential Format;
- **P**: Protocol;
- **E**: Ecosystem;
- **I**: Issuer;
- **V**: Verifier;
- **W**: Wallet.

This specification adds the requirement category as a prefix:

- **SR**: Security Requirements;

- **PR**: Privacy Requirements;
- **SPR**: Security and Privacy Requirements.

The final identifier uses the following name space: *requirement_category-component-id*.

For each requirement defined below, this section uses the description defined in [OpenID4VC-Sec Trust], specifying whether the requirement is satisfied (fully satisfied:  , partially satisfied:  , and not satisfied: ). In the following, the requirements are grouped based on their category.

15.1 Security Requirements

15.1.1 SR-CF-10 and SR-E-10



For any presentation, the Credential format and Trust Framework must be designed in a secure way to determine the Issuer and to check that the original Credential was issued by this Issuer (e.g., by using a cryptographic signature).

The IT-Wallet specification supports both SD-JWT VC and mdoc-CBOR Credential formats. The authenticity and integrity of a Credential is checked by verifying the Issuer's signature.

- For SD-JWT, the verification is performed using the algorithm specified in the **alg** header parameter of SD-JWT and a verifiable reference to the public key that must be used for the signature verification. Using OpenID Federation, the verifiable reference to the public cryptographic material is the **kid** header of the SD-JWT, where the cryptographic material is obtained from the Trust Chain about the Credential Issuer, specified in **iss** claim.
- For mdoc-CBOR, the Issuer's signature is contained in the *Mobile Security Object* (MSO) and must be validated using the Issuer's public key through a trusted certificate chain contained in the **x5chain** header parameter.

15.1.2 SR-CF-20



For any presentation, the Credential format must ensure that the data that is tied to the original Credential cannot be altered (e.g., by using a cryptographic signature.)

The cryptographic signature included in the Credential format ensures that any tampering with the Credential will result in a failed verification.

15.1.3 SR-CF-21



For cryptographic Holder binding, the presentation format requires the Holder to demonstrate possession of the private key associated with the Credential. This is typically achieved by having the Holder sign a challenge, which consists of a nonce value and the Verifier's unique identifier.

Both SD-JWT and mdoc-CBOR support cryptographic Holder binding by defining how a Holder can present a Credential to a Verifier, providing cryptographic proof of legitimate possession of the Credential.

Currently, for the remote flow, IT-Wallet supports only SD-JWT presentations. In this scenario, the KB-JWT (Key-Bound JWT) parameter is utilized to demonstrate that the Holder possesses the private key associated with the Credential. The Holder signs the KB-JWT using a **nonce** and a Verifier identifier, using the **aud** parameter, as a challenge.

15.1.4 SR-E-20



The Trust Framework must ensure that the identification of an Issuer is unique and unambiguous. If there are multiple instances of the same Issuer using the same key material, the Verifier must trust all instances equally.

The IT Wallet Trust Framework ensures that each entity (e.g., an Issuer) is uniquely identified through cryptographic keys and metadata, as distributed via a verifiable attestation, such as the OpenID Federation Entity Configuration verified within a Trust Chain.

15.1.5 SR-E-30



The way in which the Verifier determines the trustworthiness of the Issuer defined in the Trust Framework must be secured from influence by a malicious party that can, for example, introduce untrustworthy entities into a directory.

Issuers are registered by a Trust Anchor or its Intermediate. To verify the trust of an Issuer, a Verifier must verify that the Trust Chain related to the Issuer is valid and still active. This validation process ensures that only trusted entities are permitted to participate in the system, preventing the introduction of untrustworthy actors.

15.1.6 SR-E-40



The Trust Framework must ensure that there is a way for Verifiers to keep their information on trusted Issuers up to date and that there is a way to revoke trust in an Issuer.

If an Issuer's Entity Statement is revoked or unavailable, means that Issuer is no longer considered valid within the federation. This ensures that Verifiers have real-time access to the status of trusted entities and can revoke trust if necessary. However, Verifiers must actively check the Issuer's status by querying federation endpoints (i.e., the fetch endpoint for obtaining the Subordinate Statement).

15.1.7 SR-I-10



The Issuer must authenticate and identify the User properly according to the expectations of the Verifier (which may be defined in a specification, Trust Framework, or by convention).

The issuance process utilizes OAuth 2.0-based flows, specifically the Authorization Code Flow, to securely authenticate the User. Moreover, the User authentication is performed using CieID scheme with high LoA, eID Substantial Authentication with MRTD Verification or the PID/ IT-Wallet ID.

15.1.8 SR-I-20



The Issuer must only use correct and up-to-date claims about the User into the Credential where verified data is expected.

When verified data is expected, the Issuer obtains the correct and up-to-date claims from the relevant Authentic Sources, ensuring their accuracy at the time of issuance.

15.1.9 SR-I-30

The Issuer must revoke a Credential once the Issuer learns about potential abuse of the Credential.

The Issuer is the entity responsible for revoking a Credential. The specification describes several use cases that may trigger a revocation process and details the revocation flow in which the Issuer revokes Credentials at the User's request (through the Wallet Instance) after verifying possession of the Credentials.

15.1.10 SR-I-40

The Issuer must only include Holder-binding data into the Credential that is tied to the actual User (and not, e.g., include a cryptographic key under control by a third party).

The issuance process securely binds the Credential to the User as follows (see Fig. 12.5):

- Authorization (Steps 8-10): The Wallet Instance sends an authorization request, and the Issuer authenticates the User using a CieID scheme or a valid PID/IT-Wallet ID, providing the Access Token to the User.
- Proof of Key Possession (Steps 12-13, 16-17): The Wallet creates a DPoP Proof JWT, binding the Access Token to the Wallet Instance. The same key is then used later to request the Credential, ensuring continuity of ownership.
- Credential Issuance (Steps 18-21): The Credential request is verified using proof of possession, which is cryptographically bound to the User. The use of the same key in the DPoP ensures that the key material is controlled by the Wallet Instance, and not by a third party.

15.1.11 SR-I-50

(conditional w.r.t I-10) The Issuer must ensure that the Credential was stored in a secure Wallet.

A trusted Wallet Attestation ensures that the Wallet Instance is secure and meets the required security standards before any Credentials are issued or stored. In Steps 5-6 of Fig. 12.5, the Wallet Instance provides a Wallet Attestation, which includes a proof of possession signed with

the Wallet's private key. This attestation confirms that the Wallet Instance is genuine and has been verified by the Wallet Provider.

The Issuer verifies this attestation before allowing the Wallet to participate in the issuance process, ensuring that the Wallet adheres to specific security standards. Afterward, all cryptographic keys generated and used in the process come from this attested Wallet Instance.

Note

There is currently an open issue on this aspect (https://github.com/openid/OpenID4VCI/issues/355) in the OpenID4VCI spec.

15.1.12 SR-P-20



The protocol must ensure that no third party can interfere with the issuance process such that the Issuer issues Credentials for the third party to the User.

This requirement is addressed by secure identification of the Issuer. The ^iss^ parameter in the authorization response assures the Wallet that the response is coming from the expected Issuer and the cryptographic verification of the received tokens ensures that they were issued by the legitimate Issuer. In addition, the use of PKCE avoids injection of the code from another session to the User session.

15.1.13 SR-P-30



The protocol must ensure that the interaction between the Wallet and Verifier is protected such that no third party can interfere with the interaction by modifying the information transmitted.

The presentation process occurs through different flows, including remote and proximity. In the case of the remote flow, a combination of signed Request Objects, **nonce** usage, Trust Chain validation, Wallet Attestation, and Holder binding ensures that no third party can interfere with or modify the information transmitted between the Wallet and the Verifier. These mechanisms align with Security Requirement P-30, protecting the interaction from tampering or injection attacks.

15.1.14 SR-P-40

The protocol must ensure that the interaction between an attacker and a Verifier cannot be forwarded to and successfully completed by a User.

In the case of the same device flow, this can be prevented by using and properly checking the **nonce** value, which is created and sent by the Verifier in the authorization request. The Verifier should maintain a mapping between User sessions and the **nonce** that is expected in the flow. The Verifier should only accept a presentation if the **nonce** in the presentation matches the **nonce** that is expected for the User session. With this countermeasure, the Verifier must detect if a presentation is sent that was not bound to the User's session or if no User session exists at all, preventing the attack.

For cross-device flow the requirement is partially satisfied as the flow is vulnerable to Cross-Device Consent Phishing attacks (an attacker could initiate the presentation flow, obtain the signed Request Object and QR code, and forward it to the victim).

Some security measures are already in place, such as the use of **nonce** and state. The **nonce** ensures freshness of the request, and the state binds the flow to a unique transaction, thus reducing the opportunity for a successful attack.

Note

Other security measures are currently under evaluation in issue number [117](https://github.com/italia/eid-wallet-it-docs/issues/117), where a list of mitigations from [OAuthCrossDeviceSec] are discussed. Two examples are:

- Short Lived/Timebound QR Codes: Reducing the lifetime of the QR code (e.g., 2-3 mins) is necessary to restrict the time window available for the attacks.
- One-Time QR Codes: One-Time QR codes restrict the possibility of attacks when the same QR code is sent to multiple victims.

15.1.15 SR-P-41

The protocol must ensure that an attacker cannot successfully forward an interaction between a Wallet and a Verifier to a Verifier under the control of the attacker.

A prerequisite for a successful attack of this kind is that the attacker has access to some messages between the Wallet and the Verifier, for example, the attacker might have access to the

presentation contained in the VP Token. Given that, the proper implementation of TLS guarantees confidentiality, avoiding the leakage of the response. In addition to TLS, the existing implementation of **nonce** and audience checks in the presentation protocol should help to meet the P-41 security requirement. Regarding the **nonce** claim, the Verifier MUST check that the **nonce** value in the VP Token matches the **nonce** value that is created by the Verifier during the authorization request. Regarding the **aud** value, the Verifier must check that the audience of the presentation matches the Verifier's identifier.

15.1.16 SR-P-50



The protocol must ensure that third parties cannot interfere with the binding process.

In the issuance phase, the Holder binding happens at the Credential request to the protected Credential endpoint. This means that the attacker needs to obtain the access token first and therefore send the request to the Credential endpoint and bind the Credentials to the keys under his control. The IT-Wallet specification requires the use of a sender-constrained access token, which means that the access token binds to the device using cryptographic materials.

The second surface for the attack is related to key management. In the case of using software-based keys, it is possible to clone the keys and move them to a device under attacker control, and in the case of stealing the Credentials as well, the attacker can easily create proof of possession of the keys. IT-Wallet is less vulnerable to these attacks as it supports local internal WSCD that uses hardware-based keys. However, the lack of a certification profile that certifies the local internal WSCD against highly capable attackers (the certification for current TEE solutions on the market reaches AVA_VAN.3 at most as shown for example in this Certification Report or Global Platform site makes the requirement only partially satisfied.

15.1.17 SR-V-10



(conditional w.r.t I-50+V-20) The Verifier must ensure that the Credential is stored in a secure Wallet.

Verifier checks the Wallet Attestation during exchanges (sent with the authorization response), ensuring that it meets the security criteria required by the Verifier and is under the sole responsibility of its issuer, the trusted Wallet Provider.

Note

Currently, no explicit security and privacy measures related to this requirement are specified in [OpenID4VC-SecTrust] and it is not clearly defined what ^stored in a secure Wallet^ means. Without this detail, this requirement is considered only partially satisfied. Indeed, the Wallet Attestation guarantees that the Wallet Instance is operating on a secure, trusted device and adheres to the strict security policies set by the Wallet Provider. However, the attestation does not directly guarantee that each Credential within the Wallet is stored securely; it verifies the overall security of the Wallet environment, within which the Credentials reside. Therefore, while the attestation supports the Verifier's confidence that the Credential comes from a secure source, it is ultimately a broad assurance of the Wallet's security, rather than a specific validation of individual Credential storage.

15.1.18 SR-V-20

(conditional w.r.t. I-10) The Verifier must ensure that the Credential was issued by an Issuer that only issues Credentials to trustworthy Wallets.

By checking the trust of the Issuer, the Verifier ensures that the Credential was issued by a trusted Issuer committed to issuing Credentials only to secure Wallets (as for SR-I-50).

Note

Currently, no explicit security and privacy measures related to this requirement are specified in [OpenID4VC-SecTrust], configuring this item as something that requires further developments and clarifications.

15.1.19 SR-W-20

The Wallet must provide trustworthy and complete information about Issuers to the User.

The Wallet Instance discovers the trusted Issuers using trusted third party resources, such as the Federation API (e.g., using the Subordinate Listing Endpoint of the Trust Anchor and its Intermediates), inspecting the Issuer metadata and Trust Marks for filtering the PID Provider.

The Issuer's information is displayed to the User during the issuance process and can be subsequently read by the User as it is inside the issued Credential. In addition to the Issuer's information, the Digital Credential Type Metadata also contains information on the Authentic Source.

15.1.20 SR-W-30



The Wallet must provide trustworthy and complete information about Verifiers to the User.

The Wallet validates the Trust Chain related to the Verifier and its information is displayed to the User before the presentation.

15.2 Privacy Requirements

15.2.1 PR-CF-30



The Credential Format must ensure that there is a robust mechanism to ensure that data that is not to be released to a Verifier cannot be extracted by the Verifier (Selective Disclosure).

Both SD-JWT and mdoc-CBOR provide Selective Disclosure capability, allowing Holders to reveal only specific fields to the Verifier.

15.2.2 PR-CF-40



The Credential Format must support correlation protection.

While Selective Disclosure is a strong tool for preventing correlation, full unlinkability is not guaranteed in all cases. Issues like Verifier collusion or Issuer tracking can arise.

Tip

Batch issuance, using different key binding keys and salts for each Credential, can mitigate Verifier/Verifier and presentation unlinkability risks.

15.2.3 PR-E-60



The Trust Framework must ensure that the Issuer cannot learn where the User uses the Credential.

The Verifier performing the Trust Evaluation about the Issuer of a Credential must not release any information to the Credential Issuer about the Wallet Instance it is interacting with. Using [OID-FED] the Issuer doesn't know who is the User presenting the Credential. In addition, privacy is protected also during the check of the Credential's status. By using Status List [TOKEN-STATUS-LIST], the IT-Wallet specification ensures that while the Verifier checks the Credential's validity, the Issuer does not learn where or when the Credential is being used.

15.2.4 PR-E-70



The Trust Framework must support correlation protection.

The following mechanisms may be implemented to reduce the correlation:

- *Verifier-Verifier*: evaluation mechanisms to ensure that a Verifier requests only the information it is authorized to obtain from the Wallet. This approach minimizes data exchange and helps prevent User profiling through potential collusion between Verifiers.
- *Issuer-Verifier*: The Issuer does not require the authentication of the Verifier during the Trust Evaluation. In principle, the Issuer does not know which Verifiers the User is accessing and will avoid User activity profiling based on the Verifier's access.

Tip

OpenID Federation Trust Marks allow the definition of custom policies suitable to the satisfaction of this requirement.

15.2.5 PR-W-40



The Wallet must ask the User for meaningful consent before a Credential is used. The Wallet must provide the User the opportunity to review any data that is released to a Verifier.

After establishing trust with the Verifier, the Wallet asks for the User's consent and provides the User the opportunity to review and select the data to be presented to the Verifier.

15.2.6 PR-W-60



The Wallet must ensure that the Issuer cannot learn where the User uses the Credential.

Same as for SR-P-80.

15.2.7 PR-W-70



The Wallet must ensure that the Verifier cannot learn that the same User is using other Verifiers.

To mitigate Verifier/Verifier linkability for SD-JWT Credentials, one proposed solution is batch issuance, which involves using different cryptographic keys used in the key binding and salts for each issued Credential. However, the effectiveness of these methods has not yet been thoroughly evaluated, even in consideration of the impacts that these might result with the user experience, and is not available for IT-Wallet yet.

15.3 Security and Privacy Requirements

15.3.1 SPR-E-50



The Trust Framework must ensure that lifecycles of keys, certificates, and Credentials are designed such that the impact of a compromise is minimized.

The Credential lifecycle includes a Credential revocation mechanism based on Status List [TOKEN-STATUS-LIST] that ensures that Credentials are properly revoked when compromised or outdated.

The revocation of a Federation Entity (i.e., Issuer, Verifier, Wallet Provider) is instead possible by not issuing the corresponding Subordinate Statement about that Entity and set a short expiration of the Trust Chain, thus preventing misuse during compromise.

Tip

In addition, [OID-FED] supports a historical key endpoint to retrieve the list of expired and revoked keys, with the motivation of the revocation.

15.3.2 SPR-P-10



The protocol must ensure that no third party can read the Credential issued by the Issuer.

To mitigate this threat, the IT-Wallet specification requires the following security mechanisms in the issuance process:

- *TLS*: Used in all communication between the Wallet and the Issuer, ensuring that data in transit is encrypted and protected from interception by attackers.
- *Wallet Attestation*: Ensures that the Wallet operates on a secure, trusted device and complies with the security standards required by the Issuer, providing additional assurance that the Issuer is interacting with a legitimate Wallet Instance.
- *DPoP*: Ensures that the Holder of the access token possesses the private key associated with it, preventing attackers from reusing intercepted tokens.
- *Holder binding*: Ties the Credential to a specific Holder, ensuring that only the legitimate Holder can use a Credential to authenticate with the Issuer.
- *redirect_uri validation*: This validation ensures that the authorization response is sent to the correct and authorized endpoint, thereby preventing unauthorized interception by malicious actors. Ensuring the integrity of the **redirect_uri** is critical to avoid any manipulation or misdirection of the URI.
- *PKCE*: Avoids injection of a legit authorization **code** in another session.

15.3.3 SPR-P-60



The protocol must ensure that during an interaction with a Verifier, an attacker cannot read PII.

The IT Wallet complies with the requirements of P-60 by ensuring that all network connections are secured by TLS.

In addition, as the Authorization Response contains PII is important that it is not sent to an attacker-controlled endpoint. In the IT Wallet specification the **response_uri**, which is the endpoint where the Wallet sends the Authorization Response, is included in the signed Request Object, which is verified by the Wallet using the Verifier's public key and Trust Chain. This guarantees that the Authorization Response is being sent to the correct endpoint. Additionally, the Authorization Response is encrypted with the Verifier's public key, ensuring that only the intended recipient can decrypt and read the sensitive information, further securing the transmission.

Another endpoint to be validated is the **redirect_uri**, which is used to redirect the User back to the Verifier after the Credential presentation is complete. In the IT-Wallet specification, the **redirect_uri** is registered and validated beforehand during the Verifier onboarding using Open ID Federation. During the presentation phase, the Wallet must validate this value by verifying the trust with the Verifier according to the Section Trust Evaluation Mechanism

In order to be sure that the **redirect_uri** is received from a legit Wallet and not from the attacker, the Verifier response endpoint upon the recipient of a valid authorization response creates a fresh cryptographic value that is linked to the authorization response and attaches it to the **redirect_uri** that is sent to the Wallet. When the Verifier receives the redirect, it can extract the response code and check with its response endpoint whether the response code was associated with this Authorization Response. (See *Redirect URI* (page 510)).

15.3.4 SPR-P-70



The protocol must ensure that during an interaction with an Issuer, an attacker cannot read PII.

In the issuance process, as the Credential inside a Credential Response contains PII, it is required that the Credential is not sent to or intercepted by an attacker. To mitigate these threats, the IT-Wallet specification requires the following security mechanisms:

- *TLS*: Used in all communication between the Wallet and the Issuer, ensuring that data in transit is encrypted and protected from interception by attackers.
- *Wallet Instance Attestation*: Ensures that the Wallet operates on a secure, trusted device and complies with the security standards required by the Issuer, providing additional assurance that the Issuer is interacting with a legitimate Wallet Instance.
- *DPoP*: Ensures that the Holder of the access token possesses the private key associated with it, preventing attackers from reusing intercepted tokens.
- *Holder binding*: Ties the Credential to a specific Holder, ensuring that only the legitimate Holder can use a Credential to authenticate with the Issuer.

- *redirect_uri validation*: This validation ensures that the authorization response is sent to the correct and authorized endpoint, thereby preventing unauthorized interception by malicious actors. Ensuring the integrity of the **redirect_uri** is critical to avoid any manipulation or misdirection of the URI.
- *PKCE*: Avoids injection of a legit authorization **code** in another session.

Tip

A further security enhancement that could be applied to add an extra layer of protection for sensitive User information is the encryption of Credential responses. OpenID4VCI standard provides the option for the Wallet to request encrypted Credentials containing PII by including a **credential_response_encryption** object in its request.

Note

Currently, no explicit security and privacy measures related to this requirement are specified in [OpenID4VC-SecTrust], it remains a work in progress.

15.3.5 SPR-P-80



The protocol must ensure that the Issuer cannot learn where the User uses the Credential.

The exchange protocol does not require any interactions between Verifiers and Issuers. In addition, privacy-preserving Status List mechanism ensure that while the Verifier checks the Credential's validity, the Issuer does not learn where or when the Credential is being used.

15.3.6 SPR-W-50



The Wallet must ensure that the Credentials and private keys are protected from unauthorized access.

To prevent unauthorized access to the Wallet, it is unlocked on the User's device by entering a personal identification number (PIN) or using biometric authentication, such as fingerprint or facial recognition, based on the User's preferences and the device's capabilities. Additionally, the cryptographic keys are securely stored within the WSCD, ensuring that only the User can access them, thereby preventing unauthorized use or tampering.

CHAPTER

SIXTEEN

GENERAL LOG RETENTION POLICIES

The retention of logs is a key element for ensuring security, including fraud prevention, incident detection, system integrity, and compliance with applicable legal obligations. It MUST also be aligned with the requirements defined in ISO/IEC 27001, in particular with regard to auditability, access control, and secure storage. As long as log management may involve the processing of personal data, it also constitutes a measure of accountability under the GDPR, with implications for data minimization, storage limitation, and purpose limitation; for these aspects, reference is made to the relevant provisions of the GDPR and sector-specific regulations.

For all matters related to log handling, Wallet Providers, Credential Issuers, Authentic Sources, and Relying Parties are considered Organizational Entities.

Logs related to Wallet data exchange activities (e.g., accesses and transactions for Credential issuance and presentation) concerning the User, as the data subject, MUST be retained for a limited period for security, fraud prevention, dispute resolution, and legal obligations. Logs of Authentic Sources that document the provision of data to Credential Issuers for the issuance of Electronic Attestations of Attributes in the IT-Wallet System MUST be retained for a limited period for the same purposes.

Organizational Entities are responsible for log retention according to their respective roles. Solutions related to Wallet Providers, Credential Issuers, Authentic Sources, and Relying Parties MUST implement audit logging for the activities of administrators and service operators with access to data exchange processes and logs.

Unless specific legal obligations dictate otherwise, and with the definition of sector-specific regulation defining appropriate motivations, **the maximum retention period for data logs is 12 months**. Logs MUST be securely stored to ensure integrity and immutability.

16.1 ISO/IEC 27001 Logging General Requirements

Organizational Entities MUST implement appropriate technical and organizational measures in compliance with ISO/IEC 27001 for maintaining an information security management system for monitoring, auditing, and incident response. Key aspects of log handling under ISO/IEC 27001 include the following requirements:

- **A.12.4 Logging and Monitoring**: Organizational Entities MUST produce logs that record exceptions, faults, and information security events. When necessary for security reasons, logs SHOULD capture user activities with a level of detail that is appropriate and sufficient to meet security requirements, ensuring that the data collected is relevant and not excessive. Logs SHOULD be reviewed regularly to ensure that security incidents are identified and addressed promptly.
- **A.12.4.1 Event Logging**: Organizational Entities MUST establish procedures for logging events, including the types of events to be logged, the information to be captured, and the retention period for logs.
- **A.12.4.2 Protection of Log Information**: Log information MUST be protected against unauthorized access and tampering. This includes implementing access controls, encryption, and integrity checks to ensure that log data remains secure and reliable.
- **A.12.4.3 Administrator and Operator Logs**: Activities of system administrators and operators MUST be logged and reviewed regularly. Organizational Entities MUST detect unauthorized activities and ensure accountability for actions taken on critical systems.
- **A.12.4.4 Clock Synchronization**: All Organizational Entities' systems involved in logging MUST be synchronized to a reliable time source, ensuring accurate timestamps for correlating events across different systems and conducting effective forensic investigations.
- **A.16.1.7 Information Security Incident Management**: Organizational Entities MUST ensure that logs are available and reliable to support incident management processes.

16.1.1 Wallet Provider Log Retention Policy

Information related to the registration and management of Wallet Instances can be retained for up to 12 months after the deactivation/revocation of the Wallet or the associated User account, unless legal obligations require longer retention.

16.1.2 Credential Issuer Log Retention Policy

Credential Issuers define the retention period for Credentials based on sector-specific regulations. In the absence of specific regulations, the retention period for Credentials SHOULD NOT exceed 12 months after the date of expiration, configured at time of issuance within the metadata of the Issuer Signed part of the Credential.

16.1.3 Authentic Source Log Retention Policy

Authentic Sources are Organizational Entities for the purposes of this chapter: relevant logs cover accesses and transactions on endpoints and services used to provide Credential Issuers with the official data used to generate Electronic Attestations of Attributes in the IT-Wallet System, including—where applicable—e-services exposed via the National Digital Data Platform (PDND).

Unless specific legal obligations or sector-specific regulations require otherwise, the **12-month** maximum retention period stated in the introductory section applies to these logs. GDPR requirements on minimization, storage limitation, and lawful basis for processing remain applicable in all cases.

16.1.4 Relying Party Log Retention Policy

Relying Parties MAY retain logs related to the processing of data received from the Wallet only for the duration necessary to provide the requested service. The maximum retention period is 24 months after the conclusion of the service or the expiration date of the presented Credentials, unless legal obligations require otherwise.

- Relying Parties SHOULD NOT log Credential presentation disclosure maps, where not necessary.

16.1.5 Wallet Instances Logging Features

In addition to the requirements included in the Consolidated Regulation (EU) No 910/2014, Art 5a 4(d) and the additional information provided within the Recitals (11) and (13), the Commission Implementing Regulation (EU) 2024/2979 establishes detailed rules for the application of Consolidated Regulation (EU) No 910/2014, including logging in the core functionalities of European Digital Identity Wallets.

Regulation (EU) 2024/2979, Article 9, which define logging obligations about Wallet Solutions and data portability mechanisms.

CHAPTER

SEVENTEEN

DEFINED TERMS AND REFERENCES

17.1 Defined Terms and Acronyms

This section aligns the IT-Wallet System's terminology with the definitions provided in ARF 2.7.3 (see ARF Annex 1). For each term, the IT-Wallet definition is compared and mapped to the ARF definition, with notes on any differences or clarifications.

The terms *User*, *Trust Service*, *Trust Model*, *Trusted List*, *Trust Framework*, *Attribute*, *Electronic Attestations of Attributes Provider* or *Trust Service Provider (TSP)*, *Person Identification Data (PID)*, *Revocation List*, *Qualified Electronic Attestations of Attributes Provider* or *Qualified Trust Service Provider (QTSP)*, *Electronic Attestation of Attributes (EAA)*, are defined in the EIDAS-ARF.

Below is the description of acronyms and definitions which are useful for further insights into topics that complement the IT-Wallet System and the interacting components.

Access Certificate
: Certificate authenticating and validating the Wallet Relying Party. Aligned with ARF 2.7.3.

Access Certificate Authority
Access CA
: Certificate Authority responsible for issuing Access Certificates to PID Providers, Attestation Providers, and Relying Parties for secure interactions within the IT-Wallet ecosystem. Not present in ARF 2.7.3; specific to IT-Wallet.

Accreditation Process
: Process performed by the National Accreditation Body to accredit CABs, resulting in an accreditation certificate. Not present in ARF 2.7.3; specific to IT-Wallet.

Attributes
User Attribute
User Claim
: A set of characteristics, qualities, rights, or permissions of a person or object, or a single piece of such information. Aligned with ARF 2.7.3.

Authentic Source

Public or private entity responsible for a repository/system considered a primary source for Attributes or PID. Aligned with ARF 2.7.3.

Authentic Source Registry

AS Registry

Registry containing Authentic Sources, their declared capabilities, and available claims, used by Credential Issuers to discover and integrate authoritative data providers. Not present in ARF 2.7.3; specific to IT-Wallet.

Authentication

Electronic process confirming the identity of a person or the origin/integrity of data. Aligned with ARF 2.7.3.

Certificate Signing Request

Request sent to a CA containing the public key and identifying information for a digital certificate. Not present in ARF 2.7.3.

Certification Process

Process by Conformity Assessment Bodies to certify the Wallet Solution, including periodic technical assessments. Not present in ARF 2.7.3; specific to IT-Wallet.

Conformity Assessment Body

Accredited body competent to assess/certify Wallet Solutions or trust service providers. Aligned with ARF 2.7.3.

Credential Issuer

Issuer

Attestation Provider

Organizational Entity providing Digital Credentials to Users (may be PID Provider or (Q)EAA Provider). Aligned with ARF 2.7.3; IT-Wallet merges PID and (Q)EAA Providers under this term.

Critical Assets

Assets (e.g., cryptographic keys) whose loss would seriously impact the Wallet Unit. Aligned with ARF 2.7.3.

Cryptographic Hardware Key Tag

Unique identifier for Cryptographic Hardware Keys, used to access the private key in hardware. Not present in ARF 2.7.3.

Cryptographic Hardware Keys

Key pair generated by the Wallet Instance, valid for its lifetime. Not present in ARF 2.7.3.

Dataset_id

Alphanumeric string defined by the Authentic Source that uniquely identifies a specific

dataset relating to an Electronic Attestation of Attributes. Not present in ARF 2.7.3; specific to IT-Wallet.

Device Integrity Service

Service by device manufacturers to verify app integrity and secure key storage. Not present in ARF 2.7.3.

Digital Credential

Credential

Signed set of Attributes in a specific format (e.g., mDoc-CBOR, SD-JWT VC), may be PID or (Q)EAA. Not present in ARF 2.7.3.

Digital Credential Catalog

Electronic catalog containing information about the formats and schemes of Digital Credentials, the data contained and the Authentic Sources. The Catalog contains additional information that allows for the establishment of the authenticity and reliability of the information contained therein. Not present in ARF 2.7.3; specific to IT-Wallet.

eIDAS Trusted List

Repository of information about authoritative entities and their status used in the eIDAS trust services framework (e.g., PID Provider Trusted Lists, Wallet Provider Trusted Lists, List of Trusted List pointing to Member States QTSP Trusted Lists and Member States EAA Provider Trusted Lists, Access Certificates CA and Registration Certificates Provider Trusted Lists). Not present in ARF 2.7.3; specific to IT-Wallet.

Electronic Attestation of Attributes

Digitally verifiable attestation in electronic form, substantiating possession of attributes. Aligned with ARF 2.7.3.

Electronic Attestation of Attributes issued by or on behalf of a public sector body

Public Electronic Attestation of Attributes

Electronic Attestation of Attributes that contains Attributes deriving from a public Authentic Source. Aligned with ARF 2.7.3.

Electronic Attestation of Attributes Provider

Electronic Attestation Provider

Organizational Entity providing EAAs. Aligned with ARF 2.7.3.

Electronic Attestation of Public Interest

Credential of Public Interest

Electronic Attestation of Attributes that contains Attributes intended to certify the release, by the State or other public administrations, of authorizations, certifications, qualifications, identity and recognition documents, receipts of revenue, or to assume a fiduciary value and protection of public faith afterwards their issuance or the entries made on them and, in general, when they are considered security documents pursuant to Article 2, paragraph 10-bis, Law 13 July 1966, no. 559. Not present in ARF 2.7.3; specific to IT-Wallet.

Federation Authority

Public governance entity issuing guidelines, rules, and managing trust lists and participant status. Not present in ARF 2.7.3.

Federation Registry

Registry component of the IT-Wallet Registry Infrastructure that maintains the authoritative list of trusted entities participating in the federation, including their federation metadata, endpoints, and public keys. Not present in ARF 2.7.3; specific to IT-Wallet.

Holder

Person or entity that receives, manages, and presents Digital Credentials via the Wallet Instance. Similar with ARF 2.7.3. ARF uses this term only in Wallet-to-Wallet interactions while *Holder Wallet Unit* in other cases.

Holder Key Binding

Ability of the Holder to prove possession of the private key attested by a Trusted Third Party. Not present in ARF 2.7.3.

Identity and Access Management

Framework for managing digital identities and access to information. Not present in ARF 2.7.3.

Identity Matching

Process by which a Relying Party establishes that the person identification attributes presented in a transaction refer to the same natural person. See *Identity Matching and Identity Reconciliation* (page 538). Not present in ARF 2.7.3; specific to IT-Wallet.

Identity Reconciliation

Process by which a Relying Party, after successful identity matching, links that natural person to a previous User session or stored User record. See *Identity Matching and Identity Reconciliation* (page 538). Not present in ARF 2.7.3; specific to IT-Wallet.

Intermediate Entity

Intermediary

OpenID Federation Intermediate

Intermediate Entity as defined in OID-FED Section 1.2 within the OpenID Federation Trust Chain between a Trust Anchor and Leaf entities. This federation role is conceptually **distinct** from a **Relying Party Intermediary** under the European Digital Identity framework (EU_2024_1183); however, in the implementation profile described by these technical specifications, a **Relying Party Intermediary** is also implemented as an Open ID Federation Intermediate Entity (see **IT-Wallet Intermediary**). Aligned with ARF 2.7.3 for federation structures.

IT-Wallet ID

Electronic Attestation of Person Identification Data

An Electronic Attestation of Attributes (EAA) that contains identification data of a natural person and is issued with **national scope** only. The terms **IT-Wallet ID** and **Electronic Attestation of Person Identification Data of national scope** refer to the same national EAA. The qualifier «national scope» distinguishes this EAA from the EUDI **Person Identification Data (PID)**, which is a set of data under the European Digital Identity framework and is **not** an EAA. It enables User authentication and identification towards Relying Parties operating within the national jurisdiction. It MUST NOT be used for cross-border interactions and does **not** constitute a PID under EU_2024_1183 / EU_2024/2977. It MUST NOT be confused with the EUDI **PID**, nor with a **National Identity Provider** / national eID scheme (for example CieID / SPID). The term **National EID** MUST NOT be used as a synonym for IT-Wallet ID, to avoid confusion with those national eID schemes. Technical identifiers (`vct` and `credential_type`) are defined in the section *IT-Wallet ID Data Model* (page 392). Not present in ARF 2.7.3; specific to IT-Wallet.

IT-Wallet Intermediary

Organizational Entity that acts as a **Relying Party Intermediary** and is technically implemented as an OpenID Federation Intermediate Entity (OID-FED, Section 1.2). Therefore, in the IT-Wallet implementation profile, a Relying Party Intermediary is also an Open ID Federation Intermediate: it publishes its own Entity Configuration, issues Subordinate Statements for its affiliated Relying Parties, and issues Trust Marks to them. Within the IT-Wallet trust infrastructure, it is registered by the Trust Anchor with a specific `trust_mark_type`, which enables the Wallet Instance to identify and display to the User that the requesting Relying Party operates through a recognized Intermediary. Not present in ARF 2.7.3; specific to IT-Wallet.

IT-Wallet System

Set of Technical Solutions implementing the Italian Digital Wallet System. Not present in ARF 2.7.3; specific to IT-Wallet.

IT-Wallet System Register

Register of entities participating in the IT-Wallet System. Not present in ARF 2.7.3; specific to IT-Wallet.

Key Attestation

KA

Data object issued by a Wallet Provider that proves the keys used for key binding of Credentials reside in a trustworthy WSCD or Keystore using the Key Attestation APIs (OEM). Private keys bound to a Key Attestation MUST be stored in the same Keystore or WSCD as the Digital Credential keys that the Key Attestation attests.

Key Attestation APIs (OEM)

A device manufacturer's attestation mechanism that confirms whether cryptographic keys are stored securely in a hardware-backed keystore. Examples include Android Key Attestation API for Android devices and Apple DeviceCheck for iOS devices. Not present in ARF 2.7.3; specific to IT-Wallet.

Keystore

Local Internal hardware-backed Keystore

Hardware-backed key storage native to the User's device, such as the Trusted Execution Environment (TEE) and StrongBox on Android or the Secure Enclave on iOS. A Keystore is **not** a WSCD in the High-certifiable sense. The current implementation profile stores User keys of the Wallet Instance, of the IT-Wallet ID and of (Q)EAA in a Keystore. Not used as a defined term in ARF 2.7.3 with this restriction; specific to this implementation profile.

Level of Assurance

In the Union **electronic identification** framework, **levels of assurance** express the degree of confidence in the **correctness of the identification** of natural or legal persons and in the possibility to **rely on electronic identification means**. For **notified electronic identification schemes**, EIDAS (as amended, including the European Digital Identity Framework codified by EU_2024_1183) defines the levels **low**, **substantial**, and **high**. In these Technical Specifications, **LoA** is also used for **minimum assurance** requirements applying to **User authentication**, **Wallet Instance** context, and **Digital Credential** reliability (including catalog metadata such as minimum Level of Assurance), which MUST remain consistent with applicable Union law, national implementation of notified schemes, and the rules on **European Digital Identity Wallets** and **Person Identification Data (PID)** (including Commission Implementing Regulation (EU) 2024/2979 where relevant). Not present under this wording in ARF 2.7.3; aligned with the eIDAS / EUDI Wallet legal framework.

Member State Trusted List Provider

MS TLP

Organizational Entity designated by a Member State to compile, sign, and publish national Trusted Lists (e.g., QTSP Trusted Lists, EAA Provider Trusted Lists) and to notify their locations to the European Commission. Not present in ARF 2.7.3; specific to IT-Wallet.

Metadata

Digital artifact with information about an Organizational Entity (endpoints, public keys, etc.). Not present in ARF 2.7.3.

National Accreditation Bodies

Bodies performing accreditation under authority from a Member State. Aligned with ARF 2.7.3.

National Identity Provider

Organizational Entity providing preexisting notified electronic identification means used for User authentication (for example CieID / SPID), as distinct from the wallet-held **IT-Wallet ID** credential. Not present in ARF 2.7.3.

National Trust Anchor

Organizational Entity designated at national level to act as the root of trust for the federation, operating the national PKI and publishing authoritative Trusted Lists and federation metadata for subordinate entities (for example, Intermediaries and Leaves). Not present in ARF 2.7.3; specific to IT-Wallet.

Notification Process

Process for transferring information to the EC and inclusion in the Trusted List. Aligned with ARF 2.7.3.

Object_id

Alphanumeric string defined by the Authentic Source that uniquely identifies a specific instance of an Electronic Attestation of Attributes associated with a particular User. Not present in ARF 2.7.3; specific to IT-Wallet.

Organizational Entity

Legal person (organization or public entity) recognized to operate a role in the IT-Wallet ecosystem. Not present in ARF 2.7.3; specific to IT-Wallet.

Person Identification Data

PID

A set of data that is issued in accordance with Union or national law and that enables the establishment of the identity of a natural or legal person, or of a natural person representing another natural person or a legal person. Aligned with ARF 2.7.3 / EU_2024_1183. In these Technical Specifications, references to issuance, presentation, or revocation of a PID denote the technical PID (the signed data structure containing Person Identification Data) as used in the ARF.

PID Provider

Provider of Person Identification Data

Natural or legal person responsible for issuing and revoking the Person Identification Data and ensuring that the Person Identification Data of a User is cryptographically bound to a Wallet Unit. Aligned with ARF 2.7.3.

Policy Language

Formal language for defining security, privacy, and identity management policies. Not present in ARF 2.7.3; specific to IT-Wallet.

Primary Actors

Entities implementing Technical Solutions for the IT-Wallet System. Not present in ARF 2.7.3; specific to IT-Wallet.

Pseudonym

Alternative identifier for privacy/anonymity, allowing authentication/authorization. Aligned with ARF 2.7.3.

Qualified Electronic Attestation of Attributes

Digitally verifiable attestation issued by a QTSP, substantiating possession of attributes. Aligned with ARF 2.7.3.

Qualified Electronic Attestation of Attributes Provider

Organizational Entity providing QEAAs. Aligned with ARF 2.7.3.

Qualified Electronic Signature Provider

Trust Service Provider issuing Qualified Electronic Signature certificates. Aligned with ARF 2.7.3.

Registration Authority
Registrar

Party responsible for registering Organizational Entities by issuing Trust Assertions. Aligned with ARF 2.7.3.

Registration Certificate

Data object indicating the attributes the Relying Party has registered to request from Users. Aligned with ARF 2.7.3.

Registration Certificate Provider
Provider of Registration Certificates
Reg. Cert. Provider

Organizational Entity responsible for issuing Registration Certificates that describe the registration status and entitlements of PID Providers, Attestation Providers, and Relying Parties. Not present in ARF 2.7.3; specific to IT-Wallet.

Registration Process

Process for verifying eligibility and compliance of Organizational Entities. Aligned with ARF 2.7.3.

Relying Party
Wallet-Relying Party

Entity relying on electronic identification or Trust Service from a Wallet Instance. Aligned with ARF 2.7.3.

Relying Party Instance
Verifier App

Specific deployment of a Relying Party application or device. Aligned with ARF 2.7.3.

Relying Party Intermediary

Organizational Entity that acts on behalf of one or more Relying Parties to provide Technical Solutions for connecting to Wallet Instances and for User authentication or verification of Electronic Attestations. In Union law (EU_2024_1183, Article 5b(8)), intermediaries

acting on behalf of Relying Parties are **deemed to be Relying Parties** for registration and authentication towards European Digital Identity Wallets and **shall not store data about the content of the transaction** between the Wallet User and the intermediated Relying Party. High-level requirements for Wallet Relying Party registration, including intermediaries, minimum registration data, transparency, and common authentication mechanisms towards Wallets, are discussed in the EUDI Wallet Architecture and Reference Framework (ARF; *Topic X – Relying Party registration*, ARF_TOPIC_X_RP). IT-Wallet specifies onboarding to the national trust framework, metadata, and technical controls that implement these obligations together with applicable national implementing measures.

Relying Party Provider Backend
: Remote infrastructure with server-side components managed by a Relying Party Solution provider. Not present in ARF 2.7.3; specific to IT-Wallet.

Relying Party Solution
: Product (software/hardware/cloud) enabling Credential presentations in various contexts. Not present in ARF 2.7.3; specific to IT-Wallet.

Selective Disclosure
: Functionality enabling the User to submit a subset of Digital Credentials Data. Aligned with ARF 2.7.3.

Self-Sovereign Identity
: Approach giving individuals control over their digital identity information. Not present in ARF 2.7.3.

Signal (Signal Hub)
: A digital signal propagated through the PDND Signal Hub. It is used by Authentic Sources to notify Credential Issuers about updates to statuses and/or information within a domain managed by the Authentic Source itself.

Signal Hub
: The PDND platform managed by the PDND Manager which enables collection and distribution of Signals. It consists of two PDND e-Services: Signal Collection and Signal Distribution.

Supervision Process
: Process by a Supervisory Body to review and ensure proper functioning of the Wallet Provider and others. Not present in ARF 2.7.3; specific to IT-Wallet.

Technical Solutions
: Hardware/software systems and services implemented by Wallet Solution Providers, PID Provider, etc. Not present in ARF 2.7.3; specific to IT-Wallet.

Technical Specifications
: Specifications providing technical architecture, implementation framework, and design requirements. Aligned with ARF 2.7.3.

Trust
: Confidence in the security, reliability, and integrity of entities and their actions. Not present in ARF 2.7.3.

Trust Attestation
: Electronic attestation of compliance with the regulatory framework, cryptographically verifiable. Not present in ARF 2.7.3.

Trust Evaluation
: Process of verifying trustworthiness of registered Organizational Entities. Not present in ARF 2.7.3.

Trust Framework
: Legally enforceable set of rules and agreements for a multi-party system. Not present in ARF 2.7.3.

Trust Layer
: Architectural component enabling participants to establish trust. Not present in ARF 2. 7.3.

Trust Model
: Collection of rules ensuring legitimacy of components/entities in the IT-Wallet ecosystem. Not present in ARF 2.7.3.

Trust Relationship
: Reliable relationship between Organizational Entities after Trust Evaluation. Not present in ARF 2.7.3.

Trusted List
: Repository of information about authoritative entities and their status. Aligned with ARF 2.7.3.

User
: Natural or legal person using trust services or electronic identification means. Aligned with ARF 2.7.3.

Verifier
Credential Verifier
: A person or entity using a Relying Party Instance. Similar with ARF 2.7.3. ARF uses this term only in Wallet-to-Wallet interactions.

Wallet Instance
: Application installed on a User's device, part of the Wallet Unit, providing user interfaces. Aligned with ARF 2.7.3.

Wallet Instance Attestation
WIA
: Data object issued by a Wallet Provider proving the integrity of the Wallet Instance. This specification uses **Wallet Instance Attestation (WIA)**. The legacy name «Wallet Attestation» MUST NOT be used for this artefact. Specific to IT-Wallet.

Wallet Provider
: Organizational Entity responsible for management and provisioning of a Wallet Solution. Aligned with ARF 2.7.3.

Wallet Provider Backend
: Technical infrastructure and server-side components managed by a Wallet Provider. Not present in ARF 2.7.3.

Wallet Secure Cryptographic Application
: Application managing critical assets using cryptographic functions provided by the WSCD. Aligned with ARF 2.7.3.

Wallet Secure Cryptographic Device
: Tamper-resistant device providing an environment for the WSCA to protect critical assets. In this specification the term **WSCD** is reserved for a cryptographic device certifiable as resistant to attackers with high attack potential (for example an HSM or a smart card certified at least Common Criteria EAL4+ AVA_VAN.5). Aligned with ARF; the High-certifiable restriction is specific to this implementation profile.

Wallet Solution
: Set of Technical Solutions for the proper functioning of Wallet Instances. Aligned with ARF 2.7.3.

Wallet Unit
: Unique configuration of a Wallet Solution for an individual User, including security features. Aligned with ARF 2.7.3.

Wallet Unit Attestation
WUA
: Attestations a Wallet Unit uses during issuance of a PID or an attestation. A Wallet Unit Attestation comprises two types: Wallet Instance Attestations (WIAs), which attest the integrity of the Wallet Instance, and Key Attestations (KAs), which attest the security of cryptographic keys stored in the Wallet Unit.

Note

For any term not present in ARF 2.7.3, the IT-Wallet definition is provided as authoritative for the Italian context.

Below are the main defined terms and definitions related to User Experience aspects:

Authentication Button
: The Engagement Button that enables the User to access the Authentication process and use the services provided by Verifiers.

Brand Identity
: Collection of visual, verbal, and strategic elements that a service, a product or an entity uses to present itself to the User and to distinguish itself from others.

Brand Manual
: Document defining the rules and standards for the correct and consistent use of a Brand's distinctive elements, including visual, verbal, and application-related aspects, across different Touchpoints.

Call To Action
: A clear and direct suggestion that encourages users to take a specific action. It can be a button, a link, or another element guiding the user toward a particular goal.

Catalog
: Section of the Wallet Instance that displays the list of all the available Digital Credentials that can be obtained through the IT-Wallet Instance, and from which it is possible to start the issuing process.

Detailed View
: Extended display mode of the Digital Credentials, showing all the Attributes included.

Discovery Page
: It's the page of the Touchpoint of the Relying Party where the User lands to access their authenticated area, and it has the goal to show the User all the Authentication methods available.

Engagement Button
: Interactive element of the Interface that allows the User to trigger a process (e.g. to Authenticate, to request the issuance of a Digital Credential, etc.).

Interaction Model
: A set of characteristics that define how the User interacts with the Interface of one or multiple Touchpoints in order to complete a task or operation and achieve a specific goal.

Interface
: The set of graphic, typographical and interactive elements through which the User interacts with the Touchpoint(s) responsible for the delivery of a product or service, in compliance with [GL_DESIGN].

Preview View
: Compact visualization mode of the Digital Credential that allows it to be recognized and

distinguished in a list of Electronic Attestations thanks to the presence of minimum data or elements.

Selection Page
: It's the page of the Touchpoint of the Relying Party or of a third party offering a Credential Offer where the User can see all the Wallet Solutions available in the IT-Wallet System Register and is able to select which one to continue the process of Authentication, presentation or issuance.

Service Model
: Set of interactions between actors and touchpoints necessary for service delivery and fruition.

Touchpoint
: Point of contact (digital and not) between the User and the product or service.

Trust Mark
: A graphic element that gives evidence of the participation of the Primary Actors in the IT-Wallet System and thus guarantees adherence to its standards.

User Experience
: The set of people's perceptions and reactions resulting from the use and/or expectation of use of a product, system or service. Aligned with ISO 9241-210:2010.

Visual Identity
: Coherent set of graphic and typographic elements that visually represent a product or service and make it distinguishable and recognizable.

17.1.1 Acronyms

Below are the main acronyms used in the document:

Acronym	Description
AAL	Authenticator Assurance Level as defined in https://csrc.nist.gov/glossary/term/authenticator_assurance_level
ANPR	Anagrafe Nazionale della Popolazione Residente (Italian National Registry of the Resident Population)
API	Application Programming Interface
ARF HLR	EUDI Wallet Architecture and Reference Framework High Level Requirements
CAB	Conformity Assessment Body
CIE	Carta di Identità Elettronica (National Electronic Identity Card)
EAA	Electronic Attestation of Attributes
IAM	Identity and Access Management
LoA	Level of Assurance

continues on next page

Table 17.1 – continued from previous page

Acronym	Description
NAB	National Accreditation Body
OID4VP	OpenID for Verifiable Presentation
PDND	Piattaforma Digitale Nazionale Dati (National Digital Data Platform)
PID	Person Identification Data
PII	Personally Identifiable Information
QEAA	Qualified Electronic Attestation of Attributes
Pub-EAA	Electronic Attestation of Attributes issued by or on behalf of a public sector body
SSI	Self Sovereign Identity
VC	Verifiable Credential
VP	Verifiable Presentation
WIA	Wallet Instance Attestation
WUA	Wallet Unit Attestation
KA	Key Attestation
WSCA	Wallet Secure Cryptographic Application
WSCD	Wallet Secure Cryptographic Device

17.2 Normative References

Below the normative references and respective acronyms included in these Technical Specifications:

CAD	Legislative Decree No. 82 of March 7, 2005, as amended, containing the 'Digital Administration Code'.
REF_ACCESSIBILITY	Accessibility Guidelines for IT Tools as per Article 11 of Law 4/2004. Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services.
GL_DESIGN	Design Guidelines for websites and digital services provided by public administrations, pursuant to Article 53, paragraph 1-ter of Legislative Decree No. 82 of March 7, 2005, as amended.
EIDAS	Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC (eIDAS Regulation).

17.2.1 Official Resources

This section contains assets, resources, and artifacts that cannot be distributed as technical documentation but must be provided in their native digital format. These include source files in formats such as SVG, raster images, code repositories, configuration files, and other digital assets that are essential for the design and implementation of IT-Wallet System Technical Solutions.

Any additional technical resources, repositories, and documentation will be made available in this section in future releases of these Technical Specifications.

Official Website

The official website of the IT-Wallet System is the following address:

- **IT-Wallet Official Website**: https://www.wallet.gov.it

Graphic Assets

The following official graphic assets are available for download in the official_resources/ folder of this repository:

- **IT-Wallet Logo**: Official logo of the IT-Wallet System (SVG format) - Download the logo
- **IT-Wallet Symbol**: Official symbol of the IT-Wallet System - Download the symbol
- **IT-Wallet ID**: Official graphic element that ensures the unique identification and representation of the personal identification data Attestation (IT-Wallet ID / PID) across all Wallet Solutions - Download IT-Wallet ID
- **Trust Mark**: Official graphic element for attesting the participation in the IT-Wallet System - Download Trust Mark - Read more at section *Trust Mark* (page 17)
- **Authentication Button**: Official button for authentication via IT-Wallet - Download Authentication Button
- **Pages templates**: the models of the pages that belong to the Authentication flow: Discovery page, Selection page, QR code page, Thank you page and Error page – Go to Pages-templates/
- **Icons and pictograms**: icons and pictograms useful to implement the Authentication flow – Download OK State - Download KO state - Download order

HTML Components

The following official HTML components are available in the official_resources/ folder of this repository:

- **Authentication Button Component**: Official HTML/CSS component for authentication via IT-Wallet, based on the Bootstrap Italia library - View the component - Live demo
- **Discovery Page**: Official, self-contained HTML/CSS/JS template for selecting the authentication method (digital identity), based on the Bootstrap Italia library - View the template - Live demo
- **IT-Wallet Selection Page**: Official, self-contained HTML/CSS/JS template for selecting the digital wallet to continue authentication, based on the Bootstrap Italia library - View the template - Live demo
- **IT-Wallet Presentation QR Code Page**: Official, self-contained HTML/CSS/JS template for the OpenID4VP remote presentation **Cross Device** flow (QR code page), including a demonstrative Authorization Request payload encoded with a custom URL scheme (`haip://`, Request Object by reference, as described in *Remote Flow* (page 483)). The wallet cards on the Selection Page link to this demo. Based on the Bootstrap Italia library - View the template - Live demo

Common static assets shared by the three pages above (Bootstrap Italia, CSS, shared JavaScript, SVG sprites, favicon, and logos) live in *official_resources/shared-ui/* and are referenced with relative paths `../shared-ui/...`.

Brand Manual

The Brand Manual containing complete indications for the use of graphic assets and the visual identity of the IT-Wallet System will soon be available at the official website.

17.3 Technical References

This section details the technical references, grouped by scope, that are part of the current implementation profile. This section includes international and national standards, draft specifications, architecture references and other relevant technical documents foundational for the interoperability and the compliance with industry standards.

17.3.1 Wallet Architecture and Reference Frameworks

EIDAS-ARF	EUDI Wallet Architecture and Reference Framework (ARF) v2.8.0.

17.3.2 EUDI Wallet Standards and Technical Specifications

EUDI-TS 3	Specification of Wallet Unit Attestations (WUA) used in issuance of PID and Attestations.
EUDI-TS 10	Data Portability and Download (Export).
EUDI-TS 12	Specification of Strong Customer Authentication (SCA) Implementation with the Wallet.

17.3.3 Infrastructure of Trust

RFC 5280	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
OID-FED-WALLET	De Marco, G., Hedberg, R., Jones, M.B., Bradley, J., Dzhuvinov, V., "Open ID Federation Wallet Architectures 1.0", October 2024, Draft 03.
OID-FED	Hedberg, R., Jones, M.B., Solberg, A.A., Bradley, J., De Marco, G., Dzhuvinov, V., "OpenID Federation 1.0", February 2026, Final.
OID-FED-SUBORDINATE-EVENTS	De Marco, G., Jones, M.B., "OpenID Federation Subordinate Events Endpoint 1.0", January 2026, Draft 00.
ETSI TS 119 412-6	ETSI TS 119 412-6 v1.1.1 - Electronic Signatures and Infrastructures (ESI); Part 6: Certificate profile requirements for PID, Wallet, EAA, QEAA, and PSBEAA providers.

17.3.4 Digital Credential Data Format

SD-JWT-VC	O. Terbu, D.Fett, B. Campbell, "SD-JWT-based Verifiable Credentials (SD-JWT VC)", October 2025, Draft 12.
ISO18013-5	ISO/IEC 18013-5 2020. Information technology — Personal identification — ISO-compliant driving license — Part 5: Mobile driving license (mDL) application.

17.3.5 Digital Credential Issuance

OpenID4VCI	Lodderstedt, T., Yasuda, K., Looker, T., Bastian, P., "OpenID for Verifiable Credential Issuance 1.0", September 2025.
OPENID4VC-HAIP	K. Yasuda, Lodderstedt, T., Bormann, C., Heenan, J., "OpenID4VC High Assurance Interoperability Profile 1.0", December 2025, Final Specification.
OAUTH-ATTESTATION-CLIENT-AUTH	Looker, T., Bastian, P., Bormann, C., "OAuth 2.0 Attestation-Based Client Authentication", September 2025, Draft 7.
OAUTH-MULT-RESP-TYPE	de Medeiros, B., Scurtescu, M., Tarjan, P., Jones, M., "OAuth 2.0 Multiple Response Type Encoding Practices", February 2014.

17.3.6 Digital Credential Presentation

OpenID4VP	Terbu, O., Lodderstedt, T., Yasuda, K., Fett, D., Heenan, J., "OpenID for Verifiable Presentations 1.0", July 2025.
OAuthCrossDeviceSec	Kasselman, P., Fett, D., Skokan, F., "Cross-Device Flows: Security Best Current Practice", July 2024, Draft 8.
OPENID4VC-HAIP	K. Yasuda, Lodderstedt, T., Bormann, C., Heenan, J., "OpenID4VC High Assurance Interoperability Profile 1.0", December 2025, Final Specification.
ISO18013-5	ISO/IEC 18013-5 2020. Information technology — Personal identification — ISO-compliant driving license — Part 5: Mobile driving license (mDL) application.
OIDC-RP-Metadata	Jones, M.B., Hedberg, R., Bradley, J., Skokan, F., "OpenID Connect Relying Party Metadata Choices 1.0", March 2026.

17.3.7 Digital Credential Revocation Check Mechanisms

TOKEN-STATUS-LIST	Looker, T., Bastian, P., Bormann, C., "Token Status List (TSL)", October 2025, Draft 13.

17.3.8 National Data Interoperability Platform Specifications

MODI	"Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni", November 2023, Version 1.2.
PDND	"Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati", June 2025, Version 2.0.
OAS3	OpenAPI 3.0 API Specification.

17.3.9 National Digital Identity Platform Specifications

SPID/CIE-OpenID-Connect-Specifications	SPID/CIE OpenID Connect.

17.3.10 Security and Protection Profiles

UNI CEI EN ISO/IEC 27001:2024	Information security, cybersecurity and privacy protection — Information security management systems — Requirement

CHAPTER

EIGHTEEN

HOW TO CONTRIBUTE

The IT-Wallet project, including this document, follows an **open development process**. This approach ensures the development process is accessible to all, inviting all interested parties to participate.

Consequently, stakeholders, national and international community members are not only encouraged but also heartily welcomed to contribute to the refinement of these technical rules.

Below are several methods available for contributing to this project:

- **GitHub issues**. By opening an issue, you can seek clarification, propose enhancements, or report editorial typos. If you are working on an issue, we encourage you to open a draft pull request and link it.
- **Pull requests**. Pull requests represent active contributions to the project, typically, but not always following issue-based discussions. Once a pull request is initiated, it facilitates discussion and review of the proposed changes before they are merged into the main branch (*versione-corrente*).
- **Developers Italia Slack channel**. Slack is a messaging application designed for businesses, connecting people to the information they need. *Developers Italia* is an open community based on contributions and participation from public administrations, developers, technicians, students, and citizens. *Developers Italia* has initiated a Slack channel that [everyone can join for free](https://slack.developers.italia.it/), where you can learn about all their activities and partake in discussions.

18.1 Acknowledgements

We would like to thank the following individuals for their comments, concerns, ideas, contributions, some of which substantial, to this implementation profile and to the initial set of implementations.

- Alen Horvat
- Amir Sharif

- Andrea Misuraca
- Andrea Moro
- Andrea Prosseda
- Chris Phillips
- Chiara Leonardi
- Elisa Nicolussi Paolaz
- Emanuele De Cupis
- Emiliano Vernini
- Francesco Grauso
- Francesco Marino
- Francesco Ventola
- Gabriella Cefalù
- Giada Sciarretta
- Gian Paolo Rossi
- Giuseppe De Marco
- Klaas Wierenga
- Kristina Yasuda
- Leif Johansson
- Lorenzo Cerini
- Lorenzo Nardi
- Manuel Rafeli
- Marco Basili
- Marco Pernpruner
- Mart Aarma
- Marta Sciunnach
- Matteo Fortini
- Michele Massetti
- Michele Silletti
- Nicola Saitto

- Niels van Dijk
- Oliver Terbu
- Paul Bastien
- Pasquale Cerqua
- Pasquale De Rose
- Peter Altmann
- Riccardo Iaconelli
- Roland Hedberg
- Rosalia Galeano
- Salvatore Laiso
- Salvatore Manfredi
- Stefania Guerra
- Stefano Alifuoco
- Tahir Ahmad
- Takahiko Kawasaki
- Thomas Chiozzi
- Torsten Lodderstedt
- Valentina Carollo
- Valerio Bozzolan
- Vladimir Duzhinov
- Zahra Ebadi Ansaroudi

If anyone has been forgotten, please accept our apologies with the request to propose the modification of this page via a [Pull Request](https://github.com/italia/eid-wallet-it-docs) with a brief description of the contribution offered, during which event or channel, and during which period. We will then have the opportunity to apologize again and make amends as soon as possible, including you in the list.

CHAPTER

NINETEEN

OPEN SOURCE RELEASES

19.1 1. General Principles

The IT-Wallet ecosystem is built upon principles of openness and transparency. In accordance with the **Italian Digital Administration Code (CAD)** and the **European eIDAS 2.0 Regulation**, the adoption of open source frameworks, including during the experimentation phase, is strongly encouraged as a key enabler to ensure trust, collaboration, peer review and shared improvements across the ecosystem. This framework aligns with the **Interoperable Europe Act**, aiming to maximize public value and security through transparency.

19.2 2. Standard Requirements

All entities (Wallet Providers, Credential Issuers, Relying Parties) involved within the IT-Wallet ecosystem (hereinafter "Project Owners") **SHOULD** adhere to industry best practices to ensure the software is usable, compliant, and secure, following at least those listed in the sections below.

19.2.1 2.1 Licensing

- Project Owners **MUST** use an Open Source Initiative (OSI) approved license.
- Project Owners **SHOULD** prioritize the **EUPL-1.2** (European Union Public License version 1.2) to ensure legal compatibility within the EU public sector, or permissive licenses (e.g., Apache 2.0, MIT) for SDKs and libraries to maximize adoption.

19.2.2 2.2 Code Management & Transparency

- **Version Control**: Project Owners **MUST** use code hosting platforms that allow public access without requiring a login (e.g., GitHub, GitLab), ensuring a transparent history of changes.
- **Documentation**: Repositories **MUST** include clear documentation, such as:
  - `README.md`: Project overview and setup instructions.
  - `LICENSE.md`: Terms under which the copyright holder allows the recipient to use the software (as described in the *Licensing* section above).
  - `CONTRIBUTING.md`: Guidelines for contributions to the codebase made by externals (e. g., community contributors).
- **Community**: Project Owners **SHOULD** actively engage with the community for development and support, managing issues and requests for contributions in a timely manner.

19.2.3 2.3 Security & Quality

- **Code Quality**: Project Owners **SHOULD** implement automated testing (e.g., E2E and integration tests) and Continuous Integration pipelines. Such pipelines **SHOULD** be public to allow inspection.
- **Security Audits**: Project Owners **SHOULD** perform regular code audits and static analysis.
- **SBOM**: Project Owners **SHOULD** provide a Software Bill of Materials (SBOM) to facilitate vulnerability management, in line with the **Cyber Resilience Act (CRA)**.

19.3 3. Wallet Providers

19.3.1 3.1 Mandatory Open Source

Pursuant to the **Consolidated Regulation (EU) No 910/2014 (eIDAS 2.0)**, Art. 5a par. 3, the source code of the application software components of IT-Wallet solutions **MUST** be open-source licensed.

For duly justified security reasons, the source code of specific components, other than those installed on user devices, **MAY** remain closed, provided this does not compromise the overall auditability of the solution.

19.4 4. Credential Issuers and Relying Parties

19.4.1 4.1 Integration components

To foster ecosystem interoperability and reduce integration costs:

- Credential Issuers and Relying Parties **SHOULD** release Software Development Kits (SDKs), client libraries, and API specifications under an open-source license.

19.4.2 4.2 Backend infrastructure

Credential Issuers and Relying Parties **SHOULD** release their backend core logic under an open-source license, starting from the experimentation phase, to promote transparency and reuse.

19.5 5. Responsible Disclosure and Vulnerability Management

In compliance with the **Cyber Resilience Act (CRA)** and **NIS2 Directive (EU) 2022/2555**:

- **Security Policy**: All Open Source repositories **SHOULD** contain a `SECURITY.md` file outlining the procedure for reporting vulnerabilities.
- **Reporting**: Project Owners **SHOULD** establish a channel for responsible disclosure to handle security issues privately before public release.
- **Exploited vulnerabilities**: Project Owners **MUST** cooperate with National Computer Security Incident Response Teams (CSIRTs) regarding actively exploited vulnerabilities and strictly follow coordinated disclosure protocols to mitigate threats derived from irresponsible disclosure.

CHAPTER

TWENTY

APPENDIX

20.1 Mobile Application Instance

The Wallet and Mobile Relying Party Instances share significant similarities, with particular respect to some aspects related to initialization and integrity validation. To eliminate redundancy, this section will use the term **Mobile Application Instance** to collectively refer to both. Within this framework, the **Application Provider** assumes the responsibilities of either the Wallet Provider or the Relying Party Backend, depending on the context.

20.1.1 Mobile Application Instance Initialization

The Initialization flow enables the Mobile Application Instance to register a long-lived key pair, securely stored in an appropriate secure storage within the device, with the Application Provider. This process occurs only after the Application Provider verifies the security and Key Attestation issued by the OS manufacturer.

The flow is displayed in *Mobile Application Instance Initialization Sequence Diagram.* (page 668), while a step-by-step description is provided below.

Step 1: The User starts the Mobile Application Instance for the first time.

Step 2: The Mobile Application Instance:

- Checks whether the device meets the minimum security requirements (*WP_021* (page 739)).
- Checks if the Key Attestation APIs is available (*WP_022* (page 739)).

Note

Federation Check: The Mobile Application Instance needs to check if the Application Provider is part of the Federation, obtaining its protocol-specific Metadata

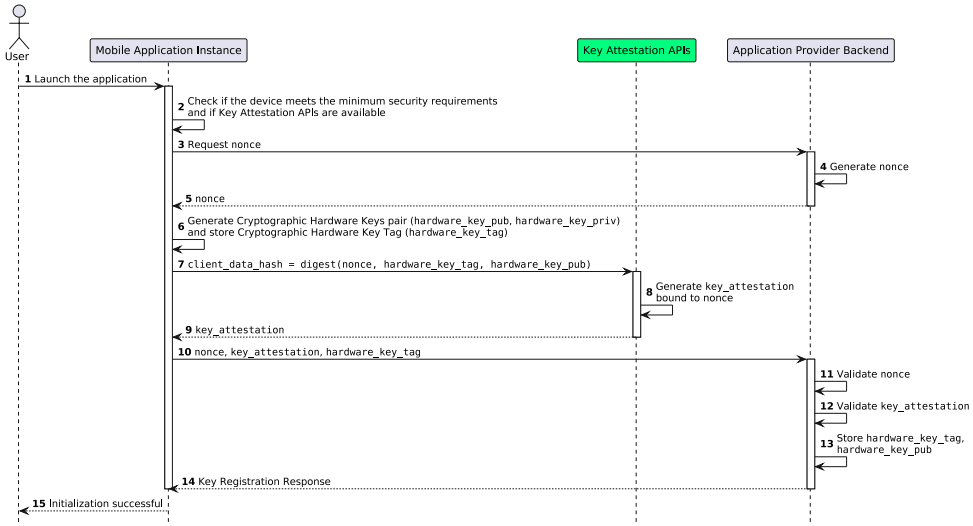


Fig. 20.1: Mobile Application Instance Initialization Sequence Diagram.

(*WP_023* (page 739)). Non-normative examples of a response from the *Federation Endpoint* (page 543) with the **Entity Configuration** and the **Metadata** of the Application Provider are presented within the *Wallet Provider Entity Configuration* (page 305) and *Relying Party Entity Configuration* (page 352) sections.

Steps 3-5 (Nonce Retrieval): The Mobile Application Instance requests a one-time `nonce` from the **Nonce Endpoint** of the Application Provider Backend (see *Wallet Solution Nonce Endpoint* (page 544) or relying-party-provider-backend-endpoint:Relying Party Provider Backend Nonce Endpoint). This `nonce` MUST be unpredictable to serve as the main defense against replay attacks (*WP_131* (page 767)).

Upon a successful request, the Application Provider generates and returns the `nonce` value to the Mobile Application Instance, as part of the *Mobile Application Nonce Response* (page 672). The Application Provider MUST ensure that it is single-use and valid only within a specific time frame.

Step 6: The Mobile Application Instance, through the operating system, creates a pair of Cryptographic Hardware Keys and stores the corresponding Cryptographic Hardware Key Tag in local storage once the following requirements are met (*WP_132* (page 767)):

1. It MUST ensure that Cryptographic Hardware Keys do not already exist. If they do exist and the Application Instance is in the initialization phase, they MUST be deleted.
2. It MUST generate a pair of asymmetric Elliptic Curve keys (`hardware_key_pub`, `hardware_key_priv`) via a local WSCD.

3. It SHOULD obtain a unique identifier Cryptographic Hardware Key Tag (`hardware_key_tag`) for the generated Cryptographic Hardware Keys from the operating system. If the operating system permits specifying a tag during the creation of keys, then a random string for the `hardware_key_tag` MUST be selected. This random value MUST be collision-resistant and unpredictable to ensure security. To achieve this, consider using a cryptographic hash function or a secure random number generator provided by the operating system or a reputable cryptographic library.
4. If the previous points are satisfied, it MUST store the `hardware_key_tag` in local storage.

Note

WSCD: The Mobile Application Instance MAY use a local WSCD for cryptographic operations, including key generation, secure storage, and cryptographic processing, on devices that support this feature. On Android devices, Strongbox is RECOMMENDED; Trusted Execution Environment (TEE) MAY be used only when Strongbox is unavailable. For iOS devices, Secure Elements (SE) MUST be used. Given that each OEM offers a distinct SDK for accessing the local WSCD, the discussion hereafter will address this topic in a general context.

If the WSCD fails during any of these operations, for example due to hardware limitations, it will raise an error response to the Mobile Application Instance. The Mobile Application Instance MUST handle these errors accordingly to ensure secure operation. Details on error handling are left to the Mobile Application Instance implementation.

Step 7: The Mobile Application Instance uses the Key Attestation APIs, providing the `client_data_hash` to acquire the Key Attestation (*WP_133b* (page 767)).

Note

Key Attestation APIs: In this section, the Key Attestation APIs is assumed to be provided by device manufacturers. This service allows the verification of a key being securely stored within the device's hardware through a signed object. Additionally, it offers verifiable proof that a specific Mobile Application Instance is authentic, unaltered, and in its original state using a specialized signed document made for this purpose.

The service also incorporates details in the signed object, such as the device type, model, app version, operating system version, bootloader status, and other relevant information to assess whether the device has been compromised. Additionally Android devices may possess the *Key Attestation API*, a feature supported by *StrongBox Keymaster* (a physical HSM installed directly on the motherboard) or by the *TEE* (Trusted Execution Environment, a secure area of the main processor). *Key Attestation* aims to provide a way to strongly determine if a key pair is hardware-backed, what the properties of the key are, and what constraints are applied to its usage. For Apple devices, the Key Attestation API is represented by *DeviceCheck*, which provides a framework and server interface to manage device-specific data securely. *DeviceCheck* is used in combination with the *Secure Enclave*, a dedicated HSM integrated into Apple's SoCs. *DeviceCheck* can be used to attest to the integrity of the device, apps, and/or encryption keys generated on the device, ensuring they were created in a secure environment like *Secure Enclave*. Developers can leverage *DeviceCheck* functionality by using the framework itself. These services, specifically developed by the manufacturer, are integrated within the Android or iOS SDKs, eliminating the need for a predefined endpoint to access them. Additionally, as they are specifically developed for mobile architecture, they do not need to be registered as Federation Entities through national registration systems. *Secure Enclave* has been available on Apple devices since the iPhone 5s (2013). For Android devices, the inclusion of **Strongbox Keymaster** may vary by manufacturer, who decides whether to include it or not.

If any errors occur in the Key Attestation APIs process, such as device integrity verification, for example, due to unavailable Key Attestation APIss, an internal error, or an invalid nonce in the integrity request, the Key Attestation APIs raise an error response. The Mobile Application Instance MUST process these errors accordingly. Details on error handling are left to the Mobile Application Instance implementation.

Step 8: The Key Attestation APIs performs the following actions:

- Creates a Key Attestation that is linked with the provided `client_data_hash` and the public key of the Application Instance Hardware.
- Incorporates information pertaining to the device's security.
- Uses an OEM private key to sign the Key Attestation, therefore verifiable with the related OEM certificate, confirming that the Cryptographic Hardware Keys are securely managed by the operating system.

Step 9 (Mobile Application Instance Initialization Request): The Mobile Application Instance sends a *Mobile Application Instance Initialization Request* (page 673) to the Application Provider, to initialize the Mobile Application Instance, identified by the Cryptographic Hardware Key public key. The request body includes the following claims: the `nonce`, Key Attestation (`key_attestation`), and Cryptographic Hardware Key Tag (`hardware_key_tag`) as *WP_133* (page 767).

Note

It is not necessary to send the Application Instance Hardware public key because it is already included in the `key_attestation`. As seen in the previous steps, the Key Attestation APIs creates a Key Attestation linked to the provided `client_data_hash` which is the digest of the Application Provider's `nonce`, the public key of the Application Instance Hardware and its Hardware Key Tag (*WP_133a* (page 767)). This process eliminates the need to send the Application Instance Hardware public key directly, as it is already included in the Key Attestation.

Steps 10-12 (Mobile Application Instance Initialization Response): The Application Provider validates the `nonce` and `key_attestation` signature (*WP_135–137* (page 767)), therefore:

1. It MUST verify that the `nonce` was generated by Application Provider and has not already been used (*WP_135a* (page 767)).
2. It MUST validate the `key_attestation` as defined by the device manufacturers' guidelines (*WP_135b* (page 767)). The Application Provider MUST also verify the binding between the received `hardware_key_tag`, `hardware_key_pub` and `nonce` with the `client_data_hash` provided in the Key Attestation (*WP_136* (page 767)).
3. It MUST verify that the device in use has no security flaws and reflects the minimum security requirements defined by the Application Provider (*WP_135b* (page 767)).
4. If these checks are passed, it MUST register the Mobile Application Instance, keeping the Cryptographic Hardware Key Tag (`hardware_key_tag`), the Public Hardware Key (`hardware_key_pub`) and possibly other useful information related to the device (*WP_137* (page 767)).

Upon successful initialization of the Mobile Application Instance, the Application Provider responds with a confirmation of success (*Mobile Application Instance Initialization Response* (page 674) and *WP_135* (page 767)).

Note

The Application Provider might associate the Mobile Application Instance (through the `hardware_key_tag` identifier) with a specific User or Device (*WP_138* (page 767)). This uniquely identifies the User/Device within the Application Provider's systems and can be used for future revocations in the lifecycle of the Mobile Application Instance.

Steps 13-14: The Mobile Application Instance has been initialized.

Note

Threat Model: while the initialization endpoint does not necessitate authenticating the client, it is safeguarded through the use of *key_attestation*. Proper validation of this attestation permits the initialization of authentic and unaltered app instances. Any other claims submitted will not undergo validation, leading the endpoint to respond with an error. Additionally, the inclusion of a nonce helps prevent replay attacks. The authenticity of both the nonce and the `hardware_key_tag` is ensured by the signature found within the `key_attestation`.

Mobile Application Nonce Request

The Nonce Request uses the HTTP GET method.

Below is a non-normative example of a Nonce Request.

```
GET /nonce HTTP/1.1
Host: application-provider.example.com
```

Mobile Application Nonce Response

Upon a successful request, the Application Provider returns an HTTP Response with a `200 OK` status code, with `Content-Type` set to `application/json`.

The Nonce Response body contains the `nonce` value.

Below is a non-normative example of a Nonce Response.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "nonce": "d2JhY2NhbG91cmVqdWFuZGFt"
}
```

Mobile Application Nonce Error Response

If any errors occur, the Application Provider returns an error response. The response uses `application/json` as the `Content-Type` and includes the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered.

Below is a non-normative example of a Nonce Error Response.

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "error": "server_error",
    "error_description": "The server encountered an unexpected error."
}
```

The following table lists HTTP Status Codes and related error codes that are supported for the error response:

HTTP Status Code	Error Code	Description
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Nonce Endpoint encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Nonce Endpoint is temporarily unavailable (e.g., due to maintenance or overload).

Mobile Application Instance Initialization Request

The Instance Initialization Request uses the HTTP POST method with `Content-Type` set to `application/json`.

The Instance Initialization Request body contains the following claims:

Claim	Description	Reference
nonce	It MUST be set to the value obtained from the Application Provider through the Nonce Endpoint.	This specification.
hardware_key_tag	The unique identifier of the **Cryptographic Hardware Keys** and encoded in `base64url`.	This specification.

continues on next page

Table 20.2 – continued from previous page

Claim	Description	Reference
key_attestation	An attestation that guarantees the secure generation, storage and usage of the key pair generated by the Mobile Application Instance. This can be an array containing a certificate chain whose leaf certificate is the Key Attestation obtained from the device **Key Attestation APIs**, signed with the device hardware key.	This specification.

Below is a non-normative example of an Instance Initialization Request.

```
POST /instance-initialization HTTP/1.1
Host: application-provider.example.com
Content-Type: application/json

{
  "nonce": "d2JhY2NhbG91cmVqdWFuZGFt",
  "key_attestation": "o2NmbXRvYXBwbGUtYXBw... redacted",
  "hardware_key_tag": "WQhyDymFKsP95iFqpzdEDWW4l7aVna2Fn4JCeWHYtbU="
}
```

Mobile Application Instance Initialization Response

If an Instance Initialization Request is successfully validated, the Application Provider provides an HTTP Response with status code `204 No Content`.

Below is a non-normative example of an Instance Initialization Response.

```
HTTP/1.1 204 No content
```

Mobile Application Instance Initialization Error Response

If any errors occur, the Application Provider returns an error response. The response uses `application/json` as the `Content-Type` and includes the following parameters:

- *error*. The error code.
- *error_description*. Text in human-readable form providing further details to clarify the nature of the error encountered (*WP_035* (page 739)).

Below is a non-normative example of an Instance Initialization Error Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store
```

```
{
    "error": "forbidden",
    "error_description": "The provided nonce is invalid, expired, or already used."
}
```

The following table lists HTTP Status Codes and related error codes that are supported for the error response (*WP_036–040* (page 739)):

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	The request is malformed, missing required parameters, or includes invalid and unknown parameters.
`403 Forbidden`	`integrity_check_error`	The device does not meet the Application Provider's minimum security requirements.
`403 Forbidden`	`invalid_request`	The provided nonce is invalid, expired, or already used.
`403 Forbidden`	`invalid_request`	The signature of the Key Attestation is invalid.
`422 Unprocessable Content` [OPTIONAL]	`validation_error`	The request does not adhere to the required format.
`500 Internal Server Error`	`server_error`	An internal error occurred while processing the request.
`503 Service Unavailable`	`temporarily_unavailable`	The service is unavailable. Please try again later.

20.2 e-Service PDND

The EIDAS-ARF framework empowers Member States to establish the interfaces, terms, and conditions governing communication between Credential Issuers and Authentic Sources. In the Italian context, interoperability is established by leveraging the following guidelines:

- "Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni" (MODI);
- "Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati" (PDND).

To leverage the PDND, entities MUST formally be subscribed becoming **Participants** (*Aderenti*). Within the PDND infrastructure, Participants MUST assume at least one of the following roles:

- **Providers** (*Erogatori*): expose e-Services to other Participants.
- **Consumers** (*Fruitori*): utilize e-Services offered by Providers within the PDND infrastructure.

Access to an e-Service requires Consumers to obtain a specific Access Token, known within the PDND infrastructure as a Voucher.

20.2.1 Requirements and Security Patterns

This specification is based on the following set of requirements:

ID	Description	Type
R1	The Consumer and the Provider have both subscribed to the PDND infrastructure.	Architectural
R2	The communication between the Consumer and the Provider MUST ensure data integrity, authenticity, non-repudiation and replay protection.	Security
R3	The Provider MAY require the Consumer to provide tracked data to complement the request. In that case, there MUST be a correlation between the tracked data and the Voucher.	Security
R4	The e-Services MUST be implemented in REST, thus SOAP protocol MUST NOT be used.	Technical
R5	The Provider MUST ensure, with a high degree of certainty, the Consumer's proof of possession of the Voucher.	Security

PDND and MODI define several security patterns designed to enhance specific security properties in interactions between Participants. This specification adopts the following applicable security patterns for the interaction between Participants:

Security Pattern	Compliant With
[REST_JWS_2021_POP] JWS POP Voucher Issuing Profile (*Annex 3 - Standards and technical details used for Voucher Authorization* [PDND]): REQUIRED. It adds a proof of possession on the Voucher. The Consumer using the Voucher to access an e-service MUST demonstrate the proof of possession of the private key whose public is attested on the Voucher.	R2, R4, R5
[ID_AUTH_CHANNEL_01] Direct Trust Transport-Level Security (*Annex 2 - Security Patterns* [MODI]): REQUIRED. It protects the communication between the Consumer and the Provider by ensuring confidentiality, integrity, identification of the Provider, and mitigation against replay attack and spoofing.	R1, R2

continues on next page

Table 20.5 – continued from previous page

Security Pattern	Compliant With
[INTEGRITY_REST_02] REST Payload Integrity in PDND (*Annex 2 - Security Patterns* [MODI]): CONDITIONAL. It ensures the integrity of the payload of the REST Consumer request and the REST Provider response, within the PDND Infrastructure. It is REQUIRED whenever the request/response carries a payload.	R2, R4
[AUDIT_REST_02] Submission of audit data within the REST request with correlation (*Annex 2 - Security Patterns* [MODI]): OPTIONAL. The Provider MAY request additional data tracked in the Consumer's domain, with a correlation between such data and the authentication method. In that case, this pattern MUST be used.	R3, R4
[ID_AUTH_REST_01] Direct Trust based on PDND via Platform-issued JWT tokens (*Annex 2 - Security Patterns* [MODI]).	R1, R2, R4, R5

Note

In the current specifications, the `REST_JWS_2021_POP` security pattern is implemented in accordance with **RFC 9449**.

Alternatively, the proof of possession MAY be attested using the `TrackingEvidence` JWT to provide additional data. In this context, `TrackingEvidence` JWT acts as a proof of possession of the PDND Voucher. Such implementation choices will be referred to as `POP_DPoP` and `POP_TPoP`, respectively.

The following security patterns defined in PDND and MODI MUST NOT be used as they do not comply with the requirements defined above:

- The following patterns can only be used when the Consumer cannot subscribe to the PDND infrastructure (i.e., the trust between the Participants needs to be established in a direct form), thus not complying with **R1**:
  - **[ID_AUTH_CHANNEL_02]** Direct Trust mutual Transport-Level Security (*Annex 2 - Security Patterns* [MODI])
  - **[ID_AUTH_REST_02]** Direct Trust based on PDND via Platform-issued JWT tokens and token/message uniqueness (*Annex 2 - Security Patterns* [MODI]).
  - **[INTEGRITY_REST_01]** REST Payload Integrity (*Annex 2 - Security Patterns* [MODI]).
- The following pattern does not provide correlation between the tracked data and the Voucher, thus not complying with **R3**:

  - **[AUDIT_REST_01]** Submission of audit data within the REST request (*Annex 2 - Security Patterns* [MODI]).

- The following patterns are based on a SOAP architecture, thus not complying with **R4**:
  - **[ID_AUTH_SOAP_01]** Direct Trust based on X.509 certificate with SOAP (*Annex 2 - Security Patterns* [MODI]).
  - **[ID_AUTH_SOAP_02]** Direct Trust based on X.509 certificate with SOAP and token/message uniqueness (*Annex 2 - Security Patterns* [MODI]).
  - **[INTEGRITY_SOAP_01]** SOAP Payload Integrity (*Annex 2 - Security Patterns* [MODI]).
- The following pattern does not guarantee proof of possession of the Voucher, thus not complying with **R5**:
  - **[REST_JWS_2021_Bearer]** JWS Bearer Voucher Issuing Profile (*Annex 3 - Standards and technical details used for Voucher Authorization* [PDND]).

Note

In case of `POP_TPoP` implementation, the Voucher is issued as a Bearer token. However, since it is accompanied by a proof of possession, it still complies with the `REST_JWS_2021_POP` security pattern rather than `REST_JWS_2021_Bearer`.

20.2.2 PDND Voucher Issuance

The PDND infrastructure defines two different types of Vouchers:

- **For e-Service**: allow Consumers to request data from an e-Service.
- **For Interoperability API**: allow Participants to request data from the Interoperability API, exposed by the PDND Infrastructure.

The two flows are described below.

PDND Voucher for e-Service

PDND Voucher for e-Services Prerequisites

The **Consumer** MUST comply with the following prerequisites:

- Has successfully subscribed to the PDND Infrastructure (as per R1).
- Has created a new *Client e-service* to interact with the intended e-Service. Upon creation, it has been assigned a `client_id` by the PDND Platform.
- Has registered a key pair associated with the *Client e-service*.
- Has requested to enroll in the intended e-Service.
- Has defined a new purpose for the e-Service. Upon definition, it has been assigned a `purposeId` by the PDND Platform.
- Has associated the *Client e-service* with the defined purpose.

The **Provider** MUST comply with the following prerequisites:

- Has successfully subscribed to the PDND Infrastructure (as per R1).
- Has created a new e-Service and published it within the PDND API Catalog.
- Has approved the Consumer's request to enroll in the e-Service.

PDND Voucher for e-Services Flow

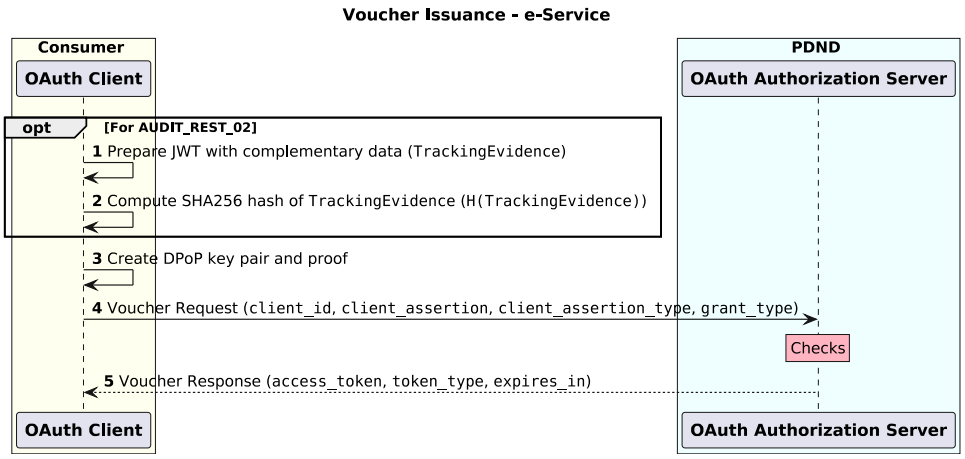


Fig. 20.2: Voucher Issuance for e-Service - Detailed flow.

Steps 1-2 (Preparation of Tracked Data): The Consumer prepares a JWT (`TrackingEvidence`) containing the tracked data that have to be sent to the Provider. Finally, it computes the SHA-256 hash of `TrackingEvidence`.

```
{
  "alg": "ES256",
  "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
  "typ": "JWT"
}
```

```
{
  "iss": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
  "aud": "https://erogatore.example/ente-example/v1",
  "exp": 1733052600,
  "nbf": 1733036450,
  "iat": 1733036400,
  "jti": "a4b5c6d7-e8f9-abcd-ef12-345678901234",
  "dnonce": 6528424213685,
  "purposeId": "b2c3d4e5-f6g7-h8i9-j0k1-lmno12345678",
  "userID": "a8b7c6d5-e4f3-g2h1-i9j0-klmnopqrstuv",
  "loa": "substantial"
}
```


Note

Steps 1-2 are required only when complying with the `AUDIT_REST_02` security pattern or the `POP_TPoP` implementation.

Step 3 (DPoP Key Pair and Proof): The Consumer MUST create a new key pair for the DPoP and a fresh DPoP proof JWT following the instruction provided in the Section 4 of **RFC 9449** for the token request to the PDND Authorization Server.

Note

Step 3 is required only when complying with the `POP_DPoP` implementation.

Step 4 (Voucher Request): The Consumer creates a Voucher Request and sends it to the PDND Authorization Server.

```
POST /authorization-server/token HTTP/1.1
Host: interop.pagopa.it
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwia2V5X29wcyI6WyJz
aWduIl0sImtpZCI6IjM5ZmE5NjBiLTc3M2YtNDllZi04YTBlLWU3NzNlOWI5N2FlOCIsImNydiI6IlAtMjU2IiwieC
I6Imh1eVhJUU52OTAyb0xzcFg0X3pvbkM5NEc2eUVsbjZsc2RtLTF3TTczMm8iLCJ5IjoiSTlQREVhd1dIcWFGREd4
MVprTmstMlBWNldkcGNhSDNBZk9iQlNMaWhndyJ9fQ.eyJqdGkiOiI1NTBlODQwMC1lMjliLTQxZDQtYTcxNi00NDY
2NTU0NDAwMDAiLCJodG0iOiJQT1NUIiwiaHR1IjoiaHR0cHM6Ly9pbnRlcm9wLnBhZ29wYS5pdC9hdXRob3JpemF0a
W9uLXNlcnZlciIsImlhdCI6MTc2MjI2MDYxNn0.D0ZUDkfGHx_rQBdYi_3VSXkdbJM-7FSWN88LWICQImMWtIWd2mb
xGb7v8udfM_c4_ase8x7I3I1JZm01Us3QEA
```

```
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_id=82914b3f-60b2-4529-b4d6-3d4e67f0a933&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2N
TQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdW
IiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJpbnRlcm9wLnBhZ29wYS5pdC9h
dXRob3JpemF0aW9uLXNlcnZlciIsImV4cCI6MTczMzA0MTQ0MCwiaWF0IjoxNzMzMDM3ODQwLCJqdGkiOiI3ZTlmM2
E0ZC1jOWIyLTQyZjYtYTZkNC0zOGUxMmZiNmI4YWIiLCJwdXJwb3NlSWQiOiJkMmI5YTY1My1jNDk3LTQ1YzYtYjhm
MS01YmRmMTI0YzlkM2EiLCJkaWdlc3QiOnsiYWxnIjoiU0hBMjU2IiwidmFsdWUiOiI5Yzc4OTRhMGE1YTkxMDU4MG
I5NjdmMzg0Y2RmYmExN2IxYWI2Zjg2NjcwZTViMGRmMThhMGM0NTNiNWViMjE1In19.cl-wvwJF3WLgywoq9qULVKC
ajleqz0jpD82QTZZAxHSjoGeA7l7n0LNC8eDfIM4F-rzMU5qfC9eW6UDxMwJrdg
```

```
{
    "alg": "ES256",
    "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
    "typ": "JWT"
}
```

```
{
    "iss": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "sub": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "aud": "https://interop.pagopa.it/authorization-server",
    "exp": 1733041440,
    "iat": 1733037840,
    "jti": "7e9f3a4d-c9b2-42f6-a6d4-38e12fb6b8ab",
    "purposeId": "d2b9a653-c497-45c6-b8f1-5bdf124c9d3a",
    "digest": {
        "alg": "SHA256",
        "value": "9c7894a0a5a910580b967f384cdfba17b1ab6f86670e5b0df18a0c453b5eb215"
    }
}
```

Note

The `purposeId` claim in the `client_assertion` payload is required only when requesting a Voucher for e-Service.

Note

The `digest` claim in the `client_assertion` payload is required only when complying with the `AUDIT_REST_02` security pattern or the `POP_TPoP` implementation.

Upon the receipt of the Voucher Request, the PDND Authorization Server MUST perform the following checks on the Voucher Request body parameters:

- The claim `client_assertion_type` is set to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- The claim `grant_type` is set to `client_credentials`.

The PDND Authorization Server MUST also validate the `client_assertion` JWT as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `JWT`.

Signature:

- Obtain the Consumer's public key corresponding to the `kid` header parameter, by interacting with the PDND Interoperability API.
- Validate the signature of the JWT using the retrieved Consumer's public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` and `sub` claims MUST identify a Client registered in the PDND Infrastructure.
- The `aud` claim MUST represent the PDND Authorization Server.
- The `exp` claim MUST represent a time instant after the current time.
- If the `nbf` claim is present, it MUST represent a time instant prior to the current time.
- The `iat` claim MUST represent a time instant prior to the current time.
- The `jti` claim MUST NOT have been previously used.
- The `purposeId` claim MUST identify a purpose registered in the PDND Infrastructure and associated to the Client.

Note

The PDND Authorization Server does not need to perform any checks on the `digest` claim.

Note

Verification of the `exp`, `nbf`, `iat`, and `jti` claims, as detailed above, MUST be performed for all JWTs described within this section. These checks will not be explicitly mentioned in subsequent references.

Step 6 (Voucher Issuance): In case of successful checks, the PDND Authorization Server issues a Voucher, which is included in the Voucher Response to the Consumer.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "access_token": "eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN
2YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjgyOTE0YjNmLT
YwYjItNDUyOS1iNGQ2LTNkNGU2N2YwYTkzMyIsImF1ZCI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1l
eGFtcGxlL3YxIiwiZXhwIjoxNzMzMDQyMTUwLCJuYmYiOjE3MzMwNDE5NDUsImlhdCI6MTczMzA0MTkyMCwianRpIj
oiYzRmNWQ3ZTItYjdjOC00MGY2LTliNmEtZGM5YTRmNWFlYjU3IiwiY2xpZW50X2lkIjoiODI5MTRiM2YtNjBiMi00
NTI5LWI0ZDYtM2Q0ZTY3ZjBhOTMzIiwicHVycG9zZUlkIjoiZDJiOWE2NTMtYzQ5Ny00NWM2LWI4ZjEtNWJkZjEyNG
M5ZDNhIiwiZGlnZXN0Ijp7ImFsZyI6IlNIQTI1NiIsInZhbHVlIjoiOTkwOGQ5NGI4ZmViMjY4YzAzNzEwNmQ3Yzg5
ZTcwNjBjMmNjMWY2YjJiNGViY2I4MDViZmVlNTNhNTM5MzA3YiJ9LCJjbmYiOnsiamt0IjoiMFpjT0NPUlpOWXktRF
dwcXEzMGpaeUpHSFROMGQySGdsQlYzdWlndUE0SSJ9fQ.sGhaHEOfMTB7r4_8ZILM_a9eTBGawWn3kL-dxYoZggFIz
yrXDOZcQWt0zr00lMk2iYAMWxS32e4cUedmAsBXGw",
    "token_type": "DPoP",
    "expires_in": 3600
}

{
    "alg": "ES256",
    "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
    "typ": "at+jwt"
}

{
    "iss": "https://interop.pagopa.it",
    "sub": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "aud": "https://erogatore.example/ente-example/v1",
    "exp": 1733042150,
    "nbf": 1733041945,
    "iat": 1733041920,
    "jti": "c4f5d7e2-b7c8-40f6-9b6a-dc9a4f5aeb57",
    "client_id": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "purposeId": "d2b9a653-c497-45c6-b8f1-5bdf124c9d3a",
    "digest": {
        "alg": "SHA256",
        "value": "9c7894a0a5a910580b967f384cdfba17b1ab6f86670e5b0df18a0c453b5eb215"
```

```
    },
    "cnf": {
        "jkt": "0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
    }
}
```

Note

The `digest` claim in the `access_token` payload is required only when complying with the `AUDIT_REST_02` security pattern or the `POP_TPoP` implementation. If present, it corresponds to the value of the `digest` claim contained in the `client_assertion`.

PDND Voucher for Interoperability API

PDND Voucher for Interoperability API Prerequisites

The **Participant** MUST comply with the following prerequisites:

- Has successfully subscribed to the PDND Infrastructure.
- Has created a new *Client interop api* to interact with the Interoperability API. Upon creation, it has been assigned a `client_id` by the PDND Platform.
- Has registered a key pair associated with the *Client interop api*.

Voucher for Interoperability API Flow

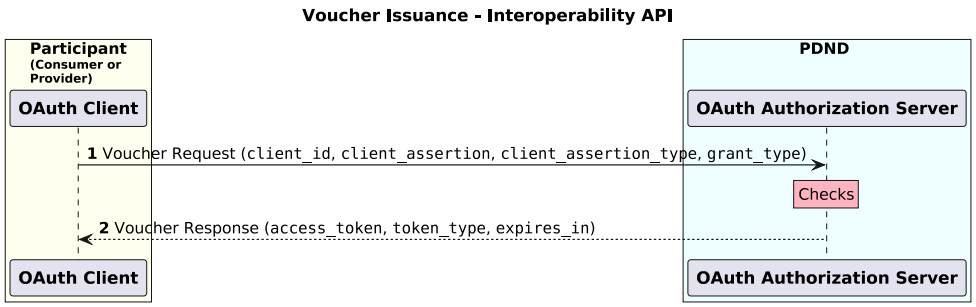


Fig. 20.3: Voucher Issuance for Interoperability API - Detailed flow.

Step 1 (Voucher Request): The Participant creates a Voucher Request and sends it to the PDND Authorization Server.

```
POST /authorization-server/token HTTP/1.1
Host: interop.pagopa.it
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_id=5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjlhNGQ4ZTNmLThiN2QtNGM5OC05MjZmLTI3NDVjNmIxZ
jgzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI1YTNjN2YyOC05MWI5LTRjNGUtODlhOS02ZTJmODVkOTI2MmIiLCJzdW
IiOiI1YTNjN2YyOC05MWI5LTRjNGUtODlhOS02ZTJmODVkOTI2MmIiLCJhdWQiOiJpbnRlcm9wLnBhZ29wYS5pdC9h
dXRob3JpemF0aW9uLXNlcnZlciIsImV4cCI6MTczMzIzMzUwMCwiaWF0IjoxNzMzMjMyMzAwLCJqdGkiOiJkMmM5YT
diNC0zZTgxLTRkMjctYjZmNy01MWE4YzlmMGEzYzYifQ.YDX7ekvvY3gPHTfZeqa3IcurU7kNBZPy3OHAdljdXSFLo
C5cVVyIzl43aMbwLouI43ylxWktaf0-pXabmye1qA

{
    "alg": "ES256",
    "kid": "9a4d8e3f-8b7d-4c98-926f-2745c6b1f832",
     "typ": "JWT"
}

{
    "iss": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
    "sub": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
    "aud": "https://interop.pagopa.it/authorization-server",
    "exp": 1733233500,
    "iat": 1733232300,
    "jti": "d2c9a7b4-3e81-4d27-b6f7-51a8c9f0a3c6"
}
```

Upon the receipt of the Voucher Request, the PDND Authorization Server MUST perform the following checks on the Voucher Request body parameters:

- The claim `client_assertion_type` is set to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- The claim `grant_type` is set to `client_credentials`.

The PDND Authorization Server MUST also validate the `client_assertion` JWT as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `JWT`.

Signature:

- Obtain the Participant's public key corresponding to the `kid` header parameter, by interacting with the PDND Interoperability API.

- Validate the signature of the JWT using the retrieved Participant's public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` and `sub` claims MUST identify a Client registered in the PDND Infrastructure.
- The `aud` claim MUST represent the PDND Authorization Server.

Step 2 (Voucher Issuance): In case of successful checks, the PDND Authorization Server issues a Voucher, which is included in the Voucher Response to the Participant.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN2Y
wOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjVhM2M3ZjI4LTkx
YjktNGM0ZS04OWE5LTZlMmY4NWQ5MjYyYiIsImF1ZCI6Imh0dHBzOi8vaW50ZXJvcC5wYWdvcGEuaXQvYXBpL3YxIi
wiZXhwIjoxNzMzMjM2NjgwLCJuYmYiOjE3MzMyMzMxNTgsImlhdCI6MTczMzIzMzA4MCwianRpIjoiZjg3ZTJkNWIt
OWY2NS00ZjBmLThhZDQtOTJlNThlNmIxM2M3IiwiY2xpZW50X2lkIjoiNWEzYzdmMjgtOTFiOS00YzRlLTg5YTktNm
UyZjg1ZDkyNjJiIn0.SKDDap16Ubi6gYwpKVdBcuhmhF_XnGiHeoxkF8F4IAualYORu_TxnDZqeP_RCcBAxSRkJTFb
MihPCLA7DoRQOw",
  "token_type": "Bearer",
  "expires_in": 3600
}

{
  "alg": "ES256",
  "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
  "typ": "at+jwt"
}
```

```
{
  "iss": "https://interop.pagopa.it",
  "sub": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
  "aud": "https://interop.pagopa.it/api/v1",
  "exp": 1733236680,
  "nbf": 1733233158,
  "iat": 1733233080,
  "jti": "f87e2d5b-9f65-4f0f-8ad4-92e58e6b13c7",
  "client_id": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b"
}
```


PDND Authorization Server Endpoint

The PDND Authorization Server Endpoint issues Vouchers to Participants. These Vouchers allow Consumers to access e-Service resources and enable Participants to interact with the Interoperability API.

PDND Voucher Request

The request to the PDND Authorization Server Endpoint adheres to the Client Credentials Grant flow specified in **RFC 6749**. The client authenticates itself by presenting a JWT-based client assertion as defined in **RFC 7521** and **RFC 7523**.

Following the specifications above, the request MUST be an HTTP POST request with a body encoded in `application/x-www-form-urlencoded` format.

The Voucher Request MUST include the following HTTP header parameters (unless otherwise specified):

Parameter	Description	Reference
DPoP	DPoP proof JWT, to comply with the `REST_JWS_2021_POP` security pattern. It is mandatory only if the requested Voucher is for e-Service (i.e., not for Interoperability API) and follows the `POP_DPoP` implementation.	[**RFC 9449**], [PDND]

The Voucher Request MUST include the following body parameters:

Parameter	Description	Reference
client_id	The unique identifier of the Participant Client, assigned by the PDND.	[**RFC 6749**], [**RFC 7521**], [**RFC 7523**], [PDND]
client_assertion	A JWT representing the client assertion.	[**RFC 7521**], [**RFC 7523**], [PDND]
client_assertion_type	MUST be set to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.	[**RFC 7521**], [**RFC 7523**]
grant_type	MUST be set to `client_credentials`.	[**RFC 6749**], [**RFC 7523**]

The `client_assertion` JWT MUST include the following JOSE header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier.	[**RFC 7515**]
kid	Unique identifier of the JWK used by the Participant to sign the `client_assertion`.	[**RFC 7515**]
typ	MUST be set to `JWT`.	[**RFC 7515**], [**RFC 7519**]

The `client_assertion` JWT MUST include the following payload claims (unless otherwise specified):

Claim	Description	Reference
iss	MUST be set to the same value as `client_id`.	[**RFC 7523**]
sub	MUST be set to the same value as `client_id`.	[**RFC 7523**]
aud	The identifier of the PDND Authorization Server Endpoint.	[**RFC 7523**]
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 7523**]
nbf	UNIX timestamp representing the JWT first validity time (optional).	[**RFC 7519**]
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 7523**]
jti	Unique identifier of the JWT to prevent replay attacks.	[**RFC 7523**]
purposeId	The identifier of the purpose registered in the PDND Platform, associated with the intended e-Service. It is mandatory only if the requested Voucher is for e-Service (i.e., not for Interoperability API).	[MODI], [PDND]

continues on next page

Table 20.8 – continued from previous page

Claim	Description	Reference
digest	JSON object containing the digest of the `TrackingEvidence` JWT. It is mandatory only if the requested Voucher is for e-Service (i.e., not for Interoperability API), and when complying with `AUDIT_REST_02` or the `POP_TPoP` implementation. If present, it MUST contain the following claims: • **alg**: JSON string representing the hashing algorithm; • **value**: JSON string representing the value of the digest.	[MODI]

PDND Voucher Response

The Voucher Response MUST include the following body parameters:

Parameter	Description	Reference
access_token	A JWT representing the access token issued by the PDND Authorization Server Endpoint.	[**RFC 6749**], [**RFC 9449**], [PDND]
token_type	It MUST be set to: • `DPoP` in case of Voucher for e-Service following the `POP_DPoP` implementation; • `Bearer` in case of Voucher for Interoperability API, or Voucher for e-Service following the `POP_TPoP` implementation.	[**RFC 6749**], [**RFC 9449**]
expires_in	Number that represents the lifetime of the access token in seconds as a positive integer.	[**RFC 6749**], [**RFC 9449**]

The `access_token` JWT MUST include the following JOSE header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier.	[**RFC 7515**]
kid	Unique identifier of the JWK used by the PDND Authorization Server Endpoint to sign the `access_token`.	[**RFC 7515**]
typ	MUST be set to `at+jwt`.	[**RFC 9068**]

The `access_token` JWT MUST include the following payload claims (unless otherwise specified):

Claim	Description	Reference
iss	The identifier of the PDND Authorization Server.	[**RFC 7519**], [**RFC 9068**]
sub	The identifier of the Participant, corresponding to the `client_id` body parameter in the Voucher Request.	[**RFC 7519**], [**RFC 9068**]
aud	The identifier of the e-Service.	[**RFC 7519**], [**RFC 9068**]
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 7519**], [**RFC 9068**]
nbf	UNIX timestamp representing the JWT first validity time (optional).	[**RFC 7519**]
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 7519**], [**RFC 9068**]
jti	Unique identifier of the JWT to prevent replay attacks.	[**RFC 7519**], [**RFC 9068**]
client_id	MUST correspond to the `client_id` contained in the Voucher Request.	[**RFC 7519**], [**RFC 8963**], [**RFC 9068**], [PDND]
purposeId	MUST correspond to the value of the `purposeId` claim contained in the Voucher Request. It is mandatory only if the requested Voucher is for e-Service (i.e., not for Interoperability API).	[MODI], [PDND]
digest	MUST correspond to the value of the `digest` object contained in the Voucher Request. It is mandatory only when complying with `AUDIT_REST_02` or the `POP_TPoP` implementation.	[MODI]
cnf	It MUST contain a **jkt** claim being JWK SHA-256 Thumbprint Confirmation Method. The value of the *jkt* member MUST be the base64url encoding (as defined in [**RFC 7515**]) of the JWK SHA-256 Thumbprint of the DPoP public key (in JWK format) to which the Access Token is bound. It is mandatory only when complying with the `POP_DPoP` implementation.	[**RFC 9449**. Section 6.1] and [**RFC 7638**].

If any errors occur during the validation of the Voucher Request, the PDND Authorization Server Endpoint MUST return an error response as defined in **RFC 6749# section-5.2**. The response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "invalid_request",
  "error_description": "The client_assertion_type parameter is missing."
}
```

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`400 Bad Request`	`invalid_request`	The request cannot be fulfilled because it is missing required parameters, contains invalid parameters, or is otherwise malformed [**RFC 6749#section-5.2**].
`400 Bad Request`	`invalid_grant`	The request cannot be fulfilled because the provided grant (i.e., `client_assertion`) is expired, revoked, already used, or otherwise malformed [**RFC 6749#section-5.2**].
`400 Bad Request`	`unsupported_grant_type`	The request cannot be fulfilled because the provided grant type is not supported by the PDND Authorization Server [**RFC 6749#section-5.2**].
`400 Bad Request`	`invalid_scope`	The request cannot be fulfilled because the provided `purposeId` is invalid, unknown, malformed, or not associated to the Client [**RFC 6749#section-5.2**].
`400 Bad Request`	`invalid_dpop_proof`	The request cannot be fulfilled because it contains an invalid *DPoP proof* [**RFC 9449#section-5**].
`401 Unauthorized`	`invalid_client`	The request cannot be fulfilled because Client Authentication failed (i.e., the `client_assertion` is malformed, incorrectly signed, missing, or unverifiable) [**RFC 6749#section-5.2**].
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the PDND Authorization Server encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the PDND Authorization Server is temporarily unavailable (e.g., due to maintenance or overload).

20.2.3 Key Retrieval

PDND Authorization Server Keys

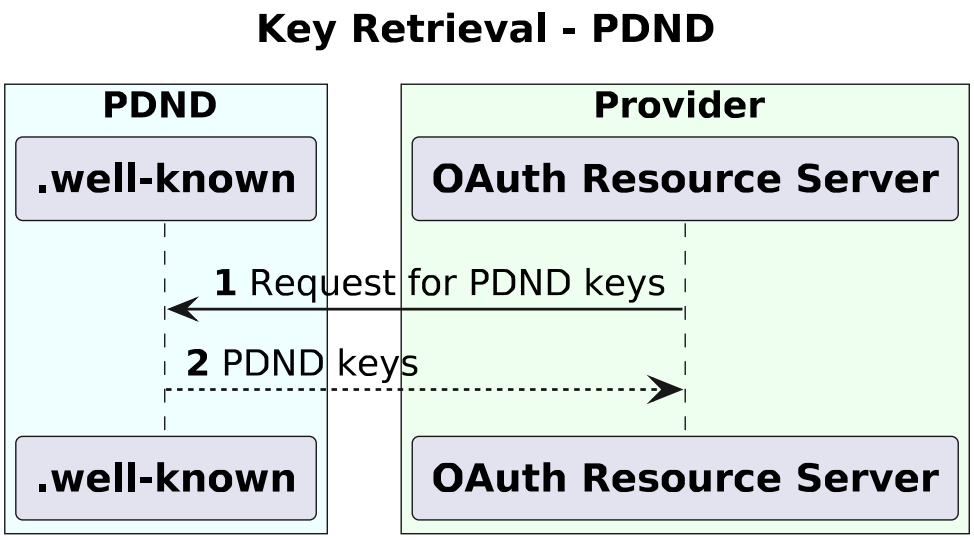


Fig. 20.4: Key Retrieval for PDND Keys - Detailed flow.

Step 1 (Keys Request): The Provider requests for the keys used by the PDND to sign Vouchers.

```
GET /.well-known/jwks.json HTTP/1.1
Host: interop.pagopa.it
```

Step 2 (Keys Response): The .well-known Endpoint returns the list of keys used by the PDND to sign Vouchers, as a `JWK Set` [RFC 7517].

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "keys": [
    {
      "kty": "RSA",
      "n": "qU2Bp7xgkXBQI2w2PZ5LZGo34TIjoir-ul0x4jZ_d9hN6q...",
      "e": "AQAB",
      "alg": "ES256",
      "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec"
    },
    {
      "kty": "RSA",
      "n": "05VukHBwiE1W_kgUS0zkOyHCrRivgw5cfSTmcvD_phieEY...",
      "e": "AQAB",
      "alg": "ES256",
```

```
      "kid": "9432c16b-7aae-49df-b9c4-ea61b556652b"
    }
  ]
}
```

PDND Authorization Server .well-known Endpoint

The .well-known Endpoint is part of the PDND Infrastructure and used to retrieve the public keys used by the PDND Authorization Server to sign the Vouchers.

PDND Authorization Server Keys Request

The Keys Request is a `GET` HTTP request sent to the .well-known Endpoint. This endpoint allows Participants to retrieve the public keys necessary to verify digital signatures on Vouchers issued by the PDND Authorization Server.

PDND Authorization Server Keys Response

The .well-known Endpoint responds with a `200 OK` status code and a `JWK Set` [RFC 7517] containing the public keys employed by the PDND Authorization Server to sign Vouchers.

If any errors occur during the retrieval of the keys, the .well-known Endpoint MUST return an error response. The response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
  "error": "server_error",
  "error_description": "The server encountered an unexpected error."
}
```

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the .well-known Endpoint encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the .well-known Endpoint is temporarily unavailable (e.g., due to maintenance or overload).

Participants' Keys

Participants' Key Retrieval Prerequisites

The **Participant** who requests the key MUST comply with the following prerequisites:

- Has successfully subscribed to the PDND Infrastructure (as per R1).
- Has created a new *Client api interop* to interact with the PDND Interoperability API. Upon creation, it has been assigned a `client_id` by the PDND Platform.
- Has registered a key pair associated with the *Client api interop*.
- Has obtained a valid Voucher to query the PDND Interoperability API, related to the specific *Client api interop*.

Participants' Key Retrieval Flow

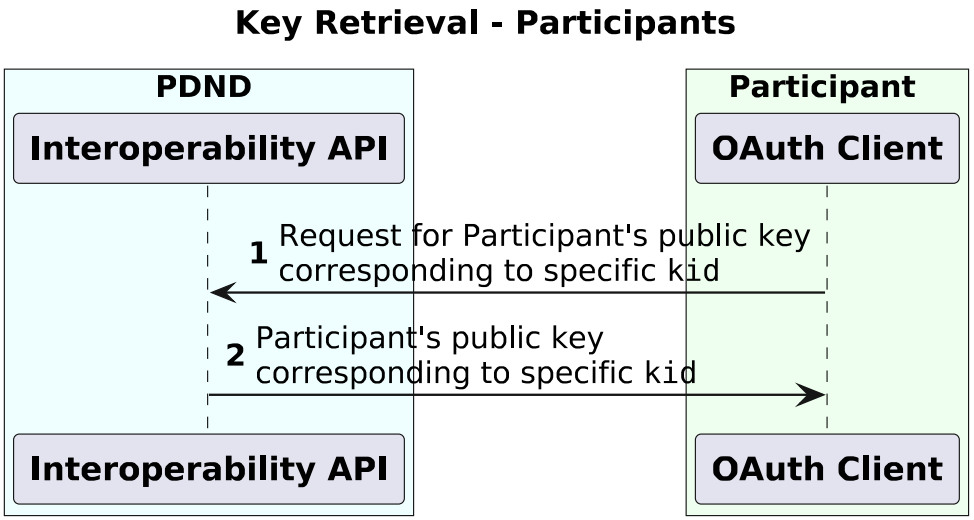


Fig. 20.5: Key Retrieval for Participant's Key - Detailed flow.

Step 1 (Key Request): The Participant requests for the key used by another Participant, corresponding to a specific `kid`, to the PDND Interoperability API.

```
GET /keys/c7e3d6a4-5b99-4298-9b84-d8f3a61279f1 HTTP/1.1
Host: interop.pagopa.it
Authorization: Bearer eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNi
N2YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjVhM2M3ZjI4L
TkxYjktNGM0ZS04OWE5LTZlMmY4NWQ5MjYyYiIsImF1ZCI6Imh0dHBzOi8vaW50ZXJvcC5wYWdvcGEuaXQvYXBpL3Y
xIiwiZXhwIjoxNzMzMjM2NjgwLCJuYmYiOjE3MzMyMzMxNTgsImlhdCI6MTczMzIzMzA4MCwianRpIjoiZjg3ZTJkN
WItOWY2NS00ZjBmLThhZDQtOTJlNThlNmIxM2M3IiwiY2xpZW50X2lkIjoiNWEzYzdmMjgtOTFiOS00YzRlLTg5YTk
tNmUyZjg1ZDkyNjJiIn0.SKDDap16Ubi6gYwpKVdBcuhmhF_XnGiHeoxkF8F4IAualYORu_TxnDZqeP_RCcBAxSRkJ
TFbMihPCLA7DoRQOw
```

Step 2 (Key Response): The Interoperability API Endpoint returns the requested key, as a `JWK` [**RFC 7517**].

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "kty": "EC",
  "key_ops": [
    "sign"
  ],
  "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
  "crv": "P-256",
  "x": "huyXIQNv902oLspX4_zonC94G6yEln6lsdm-1wM732o",
  "y": "I9PDEawWHqaFDGx1ZkNk-2PV6WdpcaH3AfObBSLihgw"
}
```

Note

The Interoperability API includes an event notification endpoint that alerts subscribed Participants about changes within the PDND Infrastructure. Among these notifications, the `/events/keys` endpoint provides updates on modifications to cryptographic material, such as additions or deletions of keys. By leveraging this mechanism, Participants can implement a periodic polling strategy to retrieve all changed keys and update their local cache. This eliminates the need to request each key individually during the workflow.

PDND Interoperability API Endpoint

The Interoperability API Endpoint is part of the PDND Infrastructure and used to retrieve the public keys of other parties enrolled in the PDND.

PDND Interoperability API Key Request

The Key Request is a `GET` HTTP request sent to the `/keys/<kid>` API. This request is used to retrieve a specific key identified by its unique `kid`.

The Key Request MUST include the following HTTP header parameters:

Parameter	Description	Reference
Authorization	Voucher released by the PDND Authorization Server.	[RFC 9449]

PDND Interoperability API Key Response

In case a public key with the provided `kid` exists, the Interoperability API Endpoint responds with a `200 OK` status code and a `JWK` [RFC 7517] representing that key.

If any errors occur during the retrieval of the key, the Interoperability API Endpoint MUST return an error response, whose structure depends on the nature of the error.

In case of authentication issues (i.e., invalid or expired Voucher), the response MUST adhere to the error format defined in RFC 6750#section-3, with specific reference to the use of the `WWW-Authenticate` header parameter.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="The access token expire
d"
```

For all other errors, the response MUST adhere to the error format defined in RFC 6749#section-5.2. The response MUST use `application/json` as the content type and MUST include the following parameters:

- `error`: The error code.
- `error_description`: Text in human-readable form providing further details to clarify the nature of the error encountered.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_request",
    "error_description": "The kid parameter is missing."
}
```

The following table lists the HTTP Status Codes and related error codes that MUST be supported for the error response:

Status Code	Error Code	Description
`400 Bad Request`	`invalid_request`	The request cannot be fulfilled because it is missing required parameters, contains invalid parameters or is otherwise malformed [RFC 6750#section-3.1].
`401 Unauthorized`	`invalid_token`	The request cannot be fulfilled because the Voucher is expired, revoked, malformed, or otherwise invalid [RFC 6750#section-3.1].
`404 Not Found`	`not_found`	The request cannot be fulfilled because no public key corresponding to the provided `kid` has been found.
`500 Internal Server Error`	`server_error`	The request cannot be fulfilled because the Interoperability API Endpoint encountered an internal problem.
`503 Service Unavailable`	`temporarily_unavailable`	The request cannot be fulfilled because the Interoperability API Endpoint is temporarily unavailable (e.g. , due to maintenance or overload).

20.2.4 e-Service Usage

e-Service Usage Prerequisites

The **Consumer** MUST comply with the following prerequisites:

- Has obtained a valid Voucher to interact with the intended e-Service, related to a specific *Client e-service*.

The **Provider** MUST comply with the following prerequisites:

- Has created a new key ring associated with the specific e-Service.
- Has registered a key pair associated with the key ring.

Note

The Provider's key ring is the counterpart to the Client on the Consumer's side. It stores cryptographic material, allowing Consumers to verify the integrity of responses from Providers.

e-Service Usage Flow

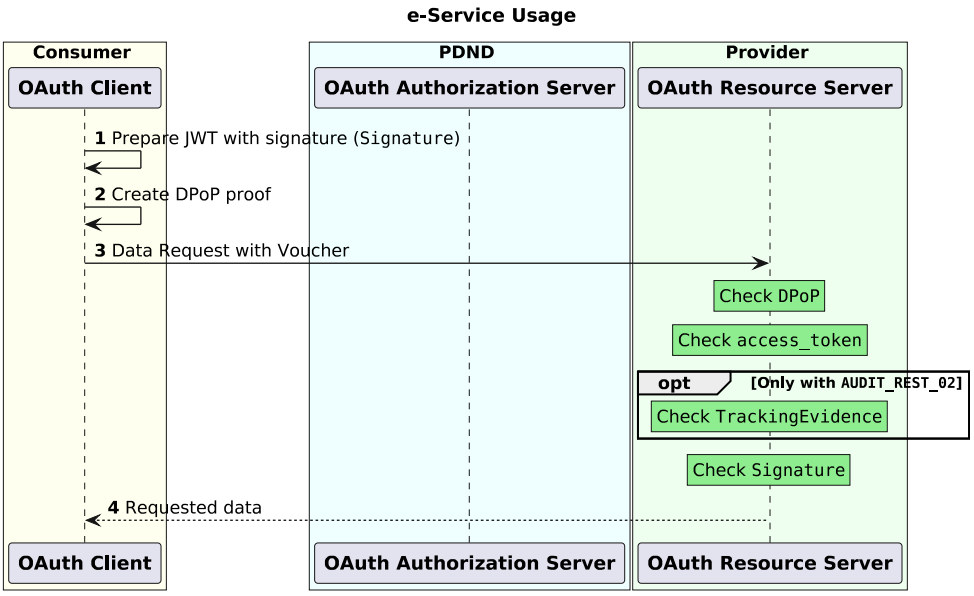


Fig. 20.6: e-Service Usage - Detailed flow.

Step 1 (Signature Preparation): The Consumer prepares a JWT (`Signature`) containing the signed headers of the message, to ensure integrity.

```
{
    "alg": "ES256",
    "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
    "typ": "JWT"
}
```

```
{
    "iss": "9a8b7c6d-e5f4-g3h2-i1j0-klmnopqrstuv",
    "sub": "9a8b7c6d-e5f4-g3h2-i1j0-klmnopqrstuv",
    "aud": "https://erogatore.example/ente-example/v1",
    "iat": 1733397840,
    "nbf": 1733401628,
    "exp": 1733401440,
    "jti": "d3f7b2c9-274a-42b7-8f8d-2e9d8b1734b0",
    "signed_headers": [
        {"digest": "SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f
41"},
        {"content-type": "application/json"}
```

```
    ]
}
```

> **Note**
>
> Step 1 is required to comply with the `INTEGRITY_REST_02` security pattern.

Step 2 (DPoP Proof for e-Service Endpoint): The Consumer MUST create a fresh DPoP Proof JWT following the instruction provided in the Section 4 of [**RFC 9449**] for the token presentation to the e-Service Endpoint.

> **Note**
>
> Step 2 is required only when complying with the `POP_DPoP` implementation.

Step 3 (e-Service Request): The Consumer sends an e-Service Request to the Provider, including the Voucher.

```
POST /ente-example/v1/hello/echo/ HTTP/1.1
Host: erogatore.example
Authorization: DPoP eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN2
YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjgyOTE0YjNmLTY
wYjItNDUyOS1iNGQ2LTNkNGU2N2YwYTkzMyIsImF1ZCI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1le
GFtcGxlL3YxIiwiZXhwIjoxNzMzMDQyMTUwLCJuYmYiOjE3MzMwNDE5NDUsImlhdCI6MTczMzA0MTkyMCwianRpIjo
iYzRmNWQ3ZTItYjdjOC00MGY2LTliNmEtZGM5YTRmNWFlYjU3IiwiY2xpZW50X2lkIjoiODI5MTRiM2YtNjBiMi00N
TI5LWI0ZDYtM2Q0ZTY3ZjBhOTMzIiwicHVycG9zZUlkIjoiZDJiOWE2NTMtYzQ5Ny00NWM2LWI4ZjEtNWJkZjEyNGM
5ZDNhIiwiZGlnZXN0Ijp7ImFsZyI6IlNIQTI1NiIsInZhbHVlIjoiOTkwOGQ5NGI4ZmViMjY4YzAzNzEwNmQ3Yzg5Z
TcwNjBjMmNjMWY2YjJiNGViY2I4MDViZmVlNTNhNTM5MzA3YiJ9LCJjbmYiOnsiamt0IjoiMFpjT0NPUlpOWXktRFd
wcXEzMGpaeUpHSFROMGQySGdsQlYzdWlndUE0SSJ9fQ.sGhaHEOfMTB7r4_8ZILM_a9eTBGawWn3kL-dxYoZggFIzy
rXDOZcQWt0zr00lMk2iYAMWxS32e4cUedmAsBXGw
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwia2V5X29wcyI6WyJz
aWduIl0sImtpZCI6IjM5ZmE5NjBiLTc3M2YtNDllZi04YTBlLWU3NzNlOWI5N2FlOCIsImNydiI6IlAtMjU2IiwieC
I6Imh1eVhJUU52OTAyb0xzcFg0X3pvbkM5NEc2eUVsbjZsc2RtLTF3TTczMm8iLCJ5IjoiSTlQREVhd1dIcWFGREd4
MVprTmstMlBWNldkcGNhSDNBZk9iQlNMaWhndyJ9fQ.eyJqdGkiOiIyYzc2ZmNhMy1jYjRlLTQzMTItOGI2ZS05NzQ
5NDYyZjQyMGQiLCJodG0iOiJQT1NUIiwiYXRoIjoiM2UwOGRlMWQwYTNkZjIzNWZjZmNjZjYyNjdmYTUwYTU5YmEyY
Tk1NTI2YzdjZTY3MDY1YjhlMjZkYmI5NDQ1MSIsImh0dSI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1
leGFtcGxlL3YxIiwiaWF0IjoxNzYyMjYyNjE2fQ.kvXh8H9B5DWCNlWyNB_PzRH217j1NHnIkE_55WnEixt2RbQTGr
CS6AFAznREA85dzqwAAaHb_qHtDc5BR0lLmQ
```

```
Agid-JWT-Signature: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5OD
c2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI5YThiN2M2ZC1lNWY0LWczaDItaTFqMC1rbG1ub3BxcnN0dXYiLCJ
zdWIiOiI5YThiN2M2ZC1lNWY0LWczaDItaTFqMC1rbG1ub3BxcnN0dXYiLCJhdWQiOiJodHRwczovL2Vyb2dhdG9yZ
S5leGFtcGxlL2VudGUtZXhhbXBsZS92MSIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE
3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZ
XJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDF
iZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.DpuBNo2U
gQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg
Digest: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f41
Agid-JWT-TrackingEvidence: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWR
jYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5
MzMiLCJhdWQiOiJodHRwczovL2Vyb2dhdG9yZS5leGFtcGxlL2VudGUtZXhhbXBsZS92MSIsImV4cCI6MTczMzA1Mj
YwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEy
LTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOG
k5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2
IiwibG9hIjoic3Vic3RhbnRpYWwifQ.bhb3f3aWEuK-bZWjyKRWrJ4hYUWhw2SQ-yRz0kUFjPQTVagjXuTqyhxsHO4
KXeSX9SivgaLSvw4n9BeZa7APbQ
Content-Type: application/json

{
    "parameter1": "value1",
    "parameter2": "value2"
}
```

The Provider MUST validate the DPoP proof [RFC 9449].

> **Note**
>
> DPoP proof validation is required only when complying with the `POP_DPoP` implementation.

The Provider MUST validate the Voucher as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `at+jwt`.

Signature:

- Retrieve the collection of public keys published at the .well-known endpoint. From this collection, select the public key whose identifier corresponds to the value of the `kid` header parameter in the Voucher.
- Validate the signature of the `access_token` using the retrieved public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` claim MUST identify the domain of the PDND Authorization Server.
- The `sub` claim MUST correspond to the `client_id` claim.
- The `aud` claim MUST match the intended e-Service.
- In case of `POP_DPoP` implementation, the `cnf.jkt` claim MUST correspond to the SHA-256 Thumbprint of the DPoP public key in the `jwk` claim in the DPoP proof.

Note

If the Provider requires additional context about the request, it can interact with the PDND Interoperability API by passing the value of the `purposeId` as a parameter.

The Provider MUST validate the `TrackingEvidence` JWT as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `JWT`.

Signature:

- Obtains the Consumer's public key corresponding to the `kid` header parameter, by interacting with the PDND Interoperability API.
- Validate the signature of the JWT using the retrieved Consumer's public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` claim MUST identify the Consumer Client.
- The `aud` claim MUST identify the Provider.

In addition, the Provider MUST ensure that the hash of the `TrackingEvidence` JWT matches the value of the `digest.value` claim contained in the `access_token` payload.

Note

The validation of the `TrackingEvidence` JWT is required only when complying with the `AUDIT_REST_02` security pattern or the `POP_TPoP` implementation.

The Provider MUST validate the `Signature` JWT as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `JWT`.

Signature:

- Validate the signature of the JWT using the retrieved Consumer's public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` and `sub` claims MUST identify the Consumer Client.
- The `aud` claim MUST identify the Provider.

In addition, the Provider MUST validate the integrity of the e-Service Request, by checking that:

- The `signed_headers.content-type` claim matches the value of the `Content-Type` HTTP header of the e-Service Request.
- The `signed_headers.digest` claim matches the value of the digest of the payload of the e-Service Request, as well as the value of the `Digest` HTTP header of the e-Service Request.

If any of the previous checks fail, the Provider MUST reject the Request.

Step 4 (e-Service Response): Upon successful checks, the Provider provides the Consumer with the requested data. In The Response header MUST be present the JWT (`Signature`) to ensure integrity.

```
HTTP/1.1 200 OK
Content-Type: application/json
Agid-JWT-Signature: ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYj
BkMi03OWUzZjBjNGE4ZWYiLA0KICAidHlwIjogIkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3
OC1pOWowLWtsbW5vcHFyc3R3eCIsDQogICJzdWIiOiAiMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdH
d4IiwNCiAgImF1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6
IDE3MzMzOTc4NDAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZXhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6IC
I4ZTEyZjRiNy05YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAic2lnbmVkX2hlYWRlcnMiOiBbDQogICAg
ew0KICAgICAgImRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTgzMDYwMGFkOWJkY2E5OTM1OTNmOD
c2MjA5YTAwNGI1OTliNTgzMDk1YjBhNjEiDQogICAgfSwNCiAgICB7DQogICAgICAiZ2VuZXJhdG9uL2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a
8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg
Digest: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b583095b0a61

{
  "requestedField1": "value1",
  "requestedField2": "value2",
  "requestedField3": "value3"
}
```

The Consumer MUST validate the `Signature` JWT as follows:

Header:

- Ensure that the `typ` claim is present and that its value is `JWT`.

Signature:

- Validate the signature of the JWT using the retrieved Provider's public key and the algorithm specified by the `alg` header parameter.

Payload:

- The `iss` and `sub` claims MUST identify the Provider.
- The `aud` claim MUST identify the Consumer Client itself.

In addition, the Consumer MUST validate the integrity of the e-Service Response, by checking that:

- The `signed_headers.content-type` claim matches the value of the `Content-Type` HTTP header of the e-Service Response.
- The `signed_headers.digest` claim matches the value of the digest of the payload of the e-Service Response, as well as the value of the `Digest` HTTP header of the e-Service Response.

If any of the previous checks fail, the Consumer MUST reject the Response.

e-Service Endpoint

e-Service Request

The e-Service Request MUST include the following HTTP header parameters (unless otherwise specified):

Parameter	Description	Reference
Authorization	Voucher released by the PDND Authorization Server.	[**RFC 9449**], [MODI], [PDND]
DPoP	DPoP proof JWT, to comply with the `REST_JWS_2021_POP` security pattern. It is mandatory only when following the `POP_DPoP` implementation.	[**RFC 9449**], [PDND]
Agid-JWT-Signature	JWT containing the signature of the message headers whose integrity needs to be guaranteed, to comply with the `INTEGRITY_REST_02` security pattern.	[MODI]

continues on next page

Table 20.15 – continued from previous page

Parameter	Description	Reference
Digest	Digest of the message payload, to comply with the `INTEGRITY_REST_02` security pattern. According to **RFC 3230**, the format MUST be the following: `<digest-algorithm>=<encoded digest output>`.	[**RFC 3230**], [MODI]
Agid-JWT-Tracking Evidence	JWT containing the data tracker in the Consumer's domain. It is mandatory only when complying with `AUDIT_REST_02` or the `POP_TPoP` implementation.	[MODI]

The `Signature` JWT, contained in the `Agid-JWT-Signature` HTTP header, MUST include the following JOSE header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier.	[**RFC 7515**]
kid	Unique identifier of the JWK used by the Consumer to sign the JWT.	[**RFC 7515**]
typ	MUST be set to `JWT`.	[**RFC 7515**], [**RFC 7519**]

The `Signature` JWT, contained in the `Agid-JWT-Signature` HTTP header, MUST include the following payload claims:

Claim	Description	Reference
iss	MUST be set to the same value as `client_id`.	[**RFC 7519**]
sub	MUST be set to the same value as `client_id`.	[**RFC 7519**]
aud	The identifier of the Provider.	[**RFC 7519**]
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 7519**]
nbf	UNIX timestamp representing the JWT first validity time (optional).	[**RFC 7519**]
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 7519**]
jti	Unique identifier of the JWT to prevent replay attacks.	[**RFC 7519**]

continues on next page

Table 20.17 – continued from previous page

Claim	Description	Reference
signed_headers	JSON object containing the signed headers whose integrity needs to be protected, to with `INTEGRITY_REST_02`. It MUST contain the following claims: • **digest**: JSON string representing the signature of the `Digest` HTTP header • **content-type**: JSON string representing the signature of the `Content-Type` HTTP header	[MODI]

If present, the `TrackingEvidence` JWT, contained in the `Agid-JWT-TrackingEvidence` HTTP header, MUST include the following JOSE header parameters:

Parameter	Description	Reference
alg	A digital signature algorithm identifier.	[**RFC 7515**]
kid	Unique identifier of the JWK used by the Consumer to sign the JWT.	[**RFC 7515**]
typ	MUST be set to `JWT`.	[**RFC 7515**], [**RFC 7519**]

If present, the `TrackingEvidence` JWT, contained in the `Agid-JWT-TrackingEvidence` HTTP header, MUST include the following payload claims:

Claim	Description	Reference
iss	MUST be set to the same value as `client_id`.	[**RFC 7519**]
aud	The identifier of the Provider.	[**RFC 7519**]
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 7519**]
nbf	UNIX timestamp representing the JWT first validity time (optional).	[**RFC 7519**]
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 7519**]
jti	Unique identifier of the JWT to prevent replay attacks.	[**RFC 7519**]
purposeId	The identifier of the purpose registered in the PDND Platform, associated with the intended e-Service.	[MODI]
dnonce	It MUST be a random string composed by integer numbers and with a length of 13 digits.	[MODI]

When complying with the `AUDIT_REST_02` security pattern, the `TrackingEvidence` payload MUST also contain the tracked data agreed upon with the Provider.

e-Service Response

The e-Service Response MUST use HTTP Content-Type set to `application/json`.

The e-Service Response MUST include the following HTTP header parameters:

Parameter	Description	Reference
Agid-JWT-Signature	JWT containing the signature of the message headers whose integrity needs to be guaranteed, to comply with the `INTEGRITY_REST_02` security pattern.	[MODI]
Digest	Digest of the message payload, to comply with the `INTEGRITY_REST_02` security pattern. According to **RFC 3230**, the format MUST be the following: `<digest-algorithm>=<encoded digest output>`.	[**RFC 3230**], [MODI]

The `Signature` JWT, contained in the `Agid-JWT-Signature` HTTP header, MUST include the following payload claims:

Claim	Description	Reference
iss	The identifier of the Provider.	[**RFC 7519**]
sub	The identifier of the Provider.	[**RFC 7519**]
aud	The identifier of the Consumer.	[**RFC 7519**]
exp	UNIX timestamp representing the JWT expiration time.	[**RFC 7519**]
nbf	UNIX timestamp representing the JWT first validity time (optional).	[**RFC 7519**]
iat	UNIX timestamp representing the JWT issuance time.	[**RFC 7519**]
jti	Unique identifier of the JWT to prevent replay attacks.	[**RFC 7519**]
signed_headers	JSON object containing the signed headers whose integrity needs to be protected, to with `INTEGRITY_REST_02`. It MUST contain the following claims: • **digest**: JSON string representing the signature of the `Digest` HTTP header • **content-type**: JSON string representing the signature of the `Content-Type` HTTP header	[MODI]

The e-Service Response payload includes specific claims related to the data elements provided to the Consumer.

If any errors occur during the validation of the e-Service Request, the e-Service Endpoint MUST return an error response, whose structure depends on the nature of the error.

In case of authentication issues (i.e., invalid or expired Voucher), the response MUST adhere to the error format defined in **RFC 6750#section-3** and **RFC 9449#section-7.1**, with specific reference to the use of the `WWW-Authenticate` header parameter.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: DPoP error="invalid_token", error_description="The access token expired"
```

For all other errors, the response MUST adhere to the error format defined in **RFC 9457**. The response MUST use `application/problem+json` (according to MODI `RAC_GEN_FORMAT_002`) as the content type and includes the following parameters:

- `type`: [OPTIONAL] URI reference that identifies the problem type.
- `title`: [REQUIRED] String containing a short, human-readable summary of the problem type.
- `status`: [REQUIRED] Number indicating the HTTP status code for this occurrence of the problem.
- `detail`: [REQUIRED] String containing a human-readable explanation specific to this occurrence of the problem.
- `instance`: [OPTIONAL] string containing a URI reference that identifies the specific occurrence of the problem.

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
    "title": "invalid_request",
    "status": 400,
    "detail": "The Agid-JWT-Signature header parameter is missing."
}
```

The following table lists the HTTP Status Codes that MUST be supported for the error response:

Status Code	Description
`400 Bad Request`	The request cannot be fulfilled because it is missing required parameters, contains invalid parameters, or is otherwise malformed.
`401 Unauthorized`	The request cannot be fulfilled because the Voucher is expired, revoked, or otherwise malformed.

continues on next page

Table 20.22 – continued from previous page

Status Code	Description
`500 Internal Server Error`	The request cannot be fulfilled because the e-Service Endpoint encountered an internal problem.
`503 Service Unavailable`	The request cannot be fulfilled because the e-Service Endpoint is temporarily unavailable (e.g., due to maintenance or overload).

20.3 PDND e-Service Template

The PDND provides a specialized tool that enhances API co-design processes by optimizing e-service publication and reuse. This functionality is defined in this document.

- "Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati" (PDND).

The template e-service serves as a standardized blueprint containing all necessary technical and descriptive metadata for an e-service. API Managers, who can be either Providers or Consumers within the PDND ecosystem, MAY create and maintain these templates.

Once a template e-service is published, it is accessible through the PDND Template Catalog, a centralized repository that facilitates discovery and reuse. This catalog enables any authorized PDND Participant to browse available templates and instantiate new e-services based on existing designs.

20.3.1 PDND Template e-service definition and guidelines

The PDND infrastructure supports the lifecycle management of Template E-Services, similar to that of traditional e-services. The lifecycle states include: **Draft**, **Active**, **Suspended**, and **Deprecated**. As with traditional e-services, PDND enforces role-based access control to govern status transitions.

Templates e-service Management

Template E-Service Creation

Participants are enabled to create Templates E-Service via a guided wizard accessible through the PDND Web GUI (APIs will be available in the future). The creation workflow closely mirrors that of standard e-service creation, with the following distinctions:

- An additional field identifies the intended recipient of the template.
- The "Audience" field is omitted.
- Thresholds are optional and serve as recommendations for Participants implementing the template.

Participants are prohibited from creating multiple templates with the same name: template names MUST be unique per participant. Upon creation, a template is initially set to the Draft state. Templates can then be published to the Template Catalog, thereby making them accessible to all Participants.

Template E-Service Modification

Participants who have created a template may edit it. The scope of editable fields depends on the template's lifecycle state:

- If the template is in Draft state, all fields are editable.
- For templates in other states, only a restricted subset of fields can be modified directly.
- Fields that cannot be modified in published templates require the creation of a new template version to apply changes.

Template versioning operates similarly to that of e-services, given that changes to the blueprint can impact instantiated services and then the Participants who consume that instance.

The following fields may be edited without triggering a new template version:

- Name
- Intended Recipient
- Description
- Voucher Time Limit
- Documentation (excluding the OpenAPI specification)
- Attributes

Template E-Service Suspension

Templates, like e-services, can be Suspended. When suspended:

- The template is removed from the templates public catalog.
- Instantiation of new instances from the suspended template is disabled.
- Previously instantiated instances remain unaffected.
- Templates may be reactivated at any time.
- Templates cannot be deleted.

Template E-Service Instantiation

Participants MAY instantiate a Template E-Service by browsing the Template Catalog and selecting a template. This process generates a new e-service.

Instantiation constraints include:

- Only templates in the Active state are eligible for instantiation.
- The instantiation is facilitated through a guided wizard in the PDND Web GUI.
- Due to the standardization objective of templates, most fields are pre-populated and immutable during instantiation.
- The following information cannot be modified during instantiation:
  - Documentation upload
  - Token expiration time
  - Name, description, and attributes

Instead, the following fields must be specified during instantiation:

- Audience
- Thresholds

- Automatic/Manual Approval Policy

Additionally, although the OpenAPI specification is fixed, the following metadata fields can be provided so that PDND can automatically update the YAML specification:

- Contacts (name, email, URL, Terms and Conditions URL)
- Server URLs

Each instantiated e-service maintains an independent lifecycle analogous to standard e-services.

Version Management

Template versioning follows a controlled process:

- Publishing a new template version sets it to Active.
- The previously Active version is automatically transitioned to Deprecated.
- Only one Active version per template is allowed at any time.
- Templates may also have a single Draft version coexisting with the Active version.

Instances derived from templates maintain independent versioning since Participants may update instance-specific fields (e.g., server URLs) multiple times, while the instance remains linked to the originating template version.

Consequently, template versions and instance versions are independent and not directly correlated.

Participants instantiating a template may then update either the specific instance or, if available, upgrade to a newer template version.

Authentic Source Template

The template e-service functionality is employed to standardize data transmission from Authentic Sources to Credential Issuers. The template e-service SHOULD be published within PDND by the Credential Issuer and is accessible through the PDND Template Catalog.

Authentic Source Template Parameters

The template e-service **MUST** adhere to the following specifications:

- **Name**: EAA Creation <Name / EAA Type Name> – IT-Wallet
- **Intended Recipients**: IT Wallet - Authentic Source - <Authentic Source domain>
- **Description**: Description text useful to the Credential Issuer about the new Credential <Credential name>
- **Technology**: REST
- **Data variation via Signal Hub**: True
- **Version changelog**: Authentic Source e-service via template implementation
- **Voucher Time Limit**: 20
- **Suggest custom threshold**: False
- **Suggest manual agreement approval policy**: False
- **Attributes**: <Official name of the Credential Issuer Public Authority>

Authentic Source Template Instantiation

Each Authentic Source **SHOULD** instantiate the *IT Wallet - Authentic Source* template e-service in PDND. The instantiation process will result in a new e-service that **MUST** satisfy the following requirements:

- **Signal Hub**: True
- **Manual agreement approval policy**: False
- **Daily API calls threshold for each provider**: greater than 10000
- **Daily API calls threshold**: greater than 10000

Additional information required during the creation process is provider-dependent.

Authentic Source PDND OpenAPI Specification

Below is the complete Open API Specification for the Authentic Source PDND e-services:

```yaml
openapi: 3.0.1
info:
  title: IT Wallet API - AS web services
  version: 0.2.0
  description: |
    # IT Wallet Authentic Source e-Service exposed via PDND.
    ### ModI patterns to be used:
      - <b>ID_AUTH_CHANNEL_01</b>: Direct Trust TLS (HTTPS)
      - <b>ID_AUTH_REST_01</b>: Authorization via PDND token (always required)
      - <b>INTEGRITY_REST_02</b>: Requests and responses are signed
      - <b>REST_JWS_2021_POP</b>: Proof of possession of the PDND Voucher (POP_DPoP per RFC 9449, or POP_TPoP)
      - <b>AUDIT_REST_02</b>: Optional. The Provider MAY require additional tracked data from the Consumer; if so, this pattern MUST be used, including together with POP_DPoP, via the optional `Agid-JWT-TrackingEvidence` header
  termsOfService: "https://authentic-source.example.it/tos/"
  contact:
    name: IT-Wallet <credential_name> <credential_provider>
    url: https://github.com/italia/eid-wallet-it-docs
  x-api-id: ASITW-01
  x-summary: IT-Wallet Authentic Source API.
servers:
  - url: https://test.authentic-source.example.it/v0.2.0
    description: Authentic Source API test server
  - url: https://authentic-source.example.it/v0.2.0
    description: Authentic Source API production server
security:
  - DPoPVoucherAuth: []
    DPoPProof: []
  - BearerAuth: []
    TrackingEvidenceAuth: []
paths:
  /status:
    get:
      tags:
        - status
      summary: Get Authentic Source API status.
      description: Health-check endpoint that returns the operational status of the Authentic Source API.
      operationId: authenticSourceStatus
      responses:
        "200":
```

```
39            description: Service available
40            content:
41              application/problem+json:
42                schema:
43                  $ref: "#/components/schemas/ProblemDetails"
44            headers:
45              Cache-Control:
46                $ref: "#/components/headers/CacheControlHeader"
47              X-RateLimit-Limit:
48                $ref: "#/components/headers/RateLimitLimitHeader"
49              X-RateLimit-Remaining:
50                $ref: "#/components/headers/RateLimitRemainingHeader"
51              X-RateLimit-Reset:
52                $ref: "#/components/headers/RateLimitResetHeader"
53          "429":
54            description: Too Many Requests
55            content:
56              application/problem+json:
57                schema:
58                  $ref: "#/components/schemas/ProblemDetails"
59            headers:
60              # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X
    -RateLimit-Reset are not required because redundant along with Retry-After.
61              Retry-After:
62                $ref: "#/components/headers/RetryAfterHeader"
63          "503":
64            description: Service Unavailable
65            content:
66              application/problem+json:
67                schema:
68                  $ref: "#/components/schemas/ProblemDetails"
69            headers:
70              Retry-After:
71                $ref: "#/components/headers/RetryAfterHeader"
72
73    /attribute-claims/{datasetId}:
74      post:
75        tags:
76          - credential
77        summary: Get Attribute Claims
78        description: >-
79          This service provides the Credential Issuer with all attribute claims necessary fo
    r the issuance of a Digital Credential
80        operationId: attributeClaims
81        parameters:
82          - in: path
83            name: datasetId
```

```
84            schema:
85              type: string
86            required: true
87            description: Identifier of the dataset as registered in the Authentic Source Reg
    istry
88          - name: Agid-JWT-Signature
89            in: header
90            description: >-
91              JWT containing the signature of the message headers whose integrity
92              needs to be guaranteed, to comply with the INTEGRITY_REST_02
93              security pattern (see <a target="blank"
94              href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-servi
    ce-pdnd.html">e-Service PDND</a>). <br/><br/>
95
96              <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIs
    ImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4M
    jkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0
    zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczM
    zM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy0
    4ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM
    2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHl
    wZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1
    VrXfKTx-CUVz_puiwqkBhulrNKj2fA">EXAMPLE
97              ON JWT.IO</a>
98            required: true
99            schema:
100             type: string
101             format: JWT
102             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5O
    Dc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLC
    JzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRp
    Yy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MD
    E0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpb
    eyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0Mj
    ZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IY
    C-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA
103         - name: Digest
104           in: header
105           description: >-
106             Digest of the message payload, to comply with the INTEGRITY_REST_02
107             security pattern. According to <a target="blank" href="https://www.rfc-editor.
    org/rfc/rfc3230.html#section-4.2">RFC
108             3230 §4.2</a>, the format MUST be the following: digest-algorithm=encoded
109             digest output.
110           required: true
111           schema:
112             type: string
```

```
113             example: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282
    f41
114         - name: Agid-JWT-TrackingEvidence
115           in: header
116           description: >-
117             TrackingEvidence JWT for AUDIT_REST_02. Optional. It MAY be sent
118             together with POP_DPoP (`Authorization: DPoP` and the `DPoP` proof)
119             when the Provider requires tracked data from the Consumer, such as
120             caller LoA, userID, IP address, or other data agreed with the
121             Provider. The DPoP proof remains the proof of possession. When the
122             Provider mandates AUDIT_REST_02, this header MUST be present.
123             For POP_TPoP the same header is required by `TrackingEvidenceAuth`
124             together with `BearerAuth`; send a single header value
125             (see <a target="blank"
126             href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-servi
    ce-pdnd.html">e-Service PDND</a>). <br/><br/>
127             <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIs
    ImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4M
    jkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2U
    uZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0a
    SI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHV
    ycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZ
    TRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8
    yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA">EXAMPLE
128             ON JWT.IO</a>
129           required: false
130           schema:
131             type: string
132             format: JWT
133             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5O
    Dc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLC
    JhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjox
    NzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMT
    IzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1u
    bzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3
    Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76c
    GBxkutDNvxRNA
134       requestBody:
135         required: true
136         content:
137           application/json:
138             schema:
139               $ref: "#/components/schemas/CredentialClaimsRequest"
140       responses:
141         "200":
142           description: OK
143           headers:
```

```
144             Agid-JWT-Signature:
145               description: JWT containing the signature of the message headers whose integ
    rity needs to be guaranteed, to comply with the INTEGRITY_REST_02 security pattern (see <a
     target="blank" href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-s
    ervice-pdnd.html">e-Service PDND</a>). <a target="blank" href="https://jwt.io/#debugger-io
    ?token=ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYjBkMi03OWUzZjB
    jNGE4ZWYiLA0KICAidHlwIjogIkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3OC1pOWowLWtsb
    W5vcHFyc3R3eCIsDQogICJzdWIiOiAiMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdHd4IiwNCiAgImF
    1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6IDE3MzMzOTc4N
    DAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZXhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6ICI4ZTEyZjRiNy0
    5YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAic2lnbmVkX2hlYWRlcnMiOiBbDQogICAgew0KICAgICAgI
    mRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTgzMDYwMGFkOWJkY2E5OTM1OTNmODc2MjA5YTAwNGI
    1OTliNTgzMDk1YjBhNjEiDQogICAgfSwNCiAgICB7DQogICAgICAiY29udGVudC10eXBlIjogImFwcGxpY2F0aW9uL
    2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a8OK1Hg0mQcZfF
    i3gelS-aL9kFS-6MoSy4csBg">EXAMPLE
146               required: true
147               schema:
148                 type: string
149             Digest:
150               description: Digest of the message payload, to comply with the INTEGRITY_RES
    T_02 security pattern. According to RFC 3230 Section 4.2 <a target="blank" href="https://w
    ww.rfc-editor.org/rfc/rfc3230.html#section-4.2">RFC 3230 §4.2</a>, the format MUST be the
    following digest-algorithm=encoded digest output.
151               required: true
152               schema:
153                 type: string
154                 example: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b5830
    95b0a61
155             Cache-Control:
156               $ref: "#/components/headers/CacheControlHeader"
157             X-RateLimit-Limit:
158               $ref: "#/components/headers/RateLimitLimitHeader"
159             X-RateLimit-Remaining:
160               $ref: "#/components/headers/RateLimitRemainingHeader"
161             X-RateLimit-Reset:
162               $ref: "#/components/headers/RateLimitResetHeader"
163           content:
164             application/json:
165               schema:
166                 $ref: "#/components/schemas/CredentialClaimsResponse"
167               example:
168                 interval: 864000
169                 userClaims:
170                   given_name: "Mario"
171                   family_name: "Rossi"
172                   birth_date: "1980-01-10"
173                   place_of_birth:
```

```
                    country: "IT"
                    region: "Roma"
                    locality: "Roma"
                  tax_id_code: "TINIT-RSSMRA80A01H501Z"
                  personal_administrative_number: "12345A123A"
                attributeClaims:
                  - object_id: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
                    status: "VALID"
                    last_updated: "2025-01-15T10:30:00Z"
                    institute_name: "Nome Istituto Universitario"
                    programme_type_name: "Laurea Magistrale"
                    degree_course_name: "Computer Science - Informatica"
                    academic_qualification_date: "2025-06-25"
                  - object_id: "7A0720AB-9C97-E122-C53E-11D05FD075GG"
                    status: "VALID"
                    last_updated: "2025-01-10T08:00:00Z"
                    institute_name: "Nome Istituto Universitario"
                    programme_type_name: "Laurea Triennale"
                    degree_course_name: "Informatica"
                    academic_qualification_date: "2022-11-27"
                metadataClaims:
                  - object_id: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
                    issuance_date: "2025-06-25"
                  - object_id: "7A0720AB-9C97-E122-C53E-11D05FD075GG"
                    issuance_date: "2022-11-27"
          "400":
            description: Bad Request
            content:
              application/problem+json:
                schema:
                  $ref: "#/components/schemas/ProblemDetails"
            headers:
              X-RateLimit-Limit:
                $ref: "#/components/headers/RateLimitLimitHeader"
              X-RateLimit-Remaining:
                $ref: "#/components/headers/RateLimitRemainingHeader"
              X-RateLimit-Reset:
                $ref: "#/components/headers/RateLimitResetHeader"
          "401":
            description: Unauthorized
            content:
              application/problem+json:
                schema:
                  $ref: "#/components/schemas/ProblemDetails"
            headers:
              X-RateLimit-Limit:
                $ref: "#/components/headers/RateLimitLimitHeader"
```

```
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
            WWW-Authenticate:
              $ref: "#/components/headers/WWWAuthenticateHeader"
        "404":
          description: Claims not found
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "429":
          description: Too Many Requests
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X-RateLimit-Reset are not required because redundant along with Retry-After.
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"
        "500":
          description: Internal Server Error.
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"
        "503":
          description: Service Unavailable
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Retry-After:
```

```yaml
              $ref: "#/components/headers/RetryAfterHeader"

tags:
  - name: status
    description: Endpoint di health check dell'API.
  - name: credential
    description: Retrieve information about the credential.

components:
  securitySchemes:
    DPoPVoucherAuth:
      type: http
      scheme: DPoP
      description: >-
        PDND Voucher issued with DPoP binding (POP_DPoP, RFC 9449). The
        `Authorization` header MUST be `DPoP <voucher>` (`token_type` is `DPoP`).
        This scheme MUST be used together with `DPoPProof`. It MUST NOT be
        modelled as HTTP Bearer: a conformant Provider rejects
        `Authorization: Bearer` for a DPoP-bound Voucher.
    DPoPProof:
      type: apiKey
      in: header
      name: DPoP
      description: >-
        DPoP proof JWT (RFC 9449). REQUIRED together with `DPoPVoucherAuth`
        (POP_DPoP). MUST NOT be sent with `BearerAuth` (POP_TPoP), where proof
        of possession is `Agid-JWT-TrackingEvidence`.
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: >-
        PDND Voucher issued as Bearer (POP_TPoP). The `Authorization` header
        MUST be `Bearer <voucher>` (`token_type` is `Bearer`). This scheme MUST
        be used together with `TrackingEvidenceAuth` as proof of possession.
        This is REST_JWS_2021_POP via TrackingEvidence, not REST_JWS_2021_Bearer.
    TrackingEvidenceAuth:
      type: apiKey
      in: header
      name: Agid-JWT-TrackingEvidence
      description: >-
        TrackingEvidence JWT. REQUIRED together with `BearerAuth` (POP_TPoP) as
        proof of possession of the PDND Voucher. The same header is also an
        optional operation parameter, so it MAY be sent with the POP_DPoP pair
        (`DPoPVoucherAuth` + `DPoPProof`) when the Provider requires
        AUDIT_REST_02. The DPoP proof remains the proof of possession.
        See e-Service PDND in the Technical Specifications.
```

```yaml

  headers:
    CacheControlHeader:
      schema:
        type: string
        enum:
          - no-store
      description: no-store
    RateLimitLimitHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: Maximum number of requests within the time window.
    RateLimitRemainingHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: Remaining requests within the time window.
    RateLimitResetHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: UTC epoch in seconds, corresponding to when the window for the current
  rate limit will reset.
    RetryAfterHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: Seconds to wait before receiving another response.
    WWWAuthenticateHeader:
      schema:
        type: string
        example: >-
          Bearer error="invalid_token", error_description="The access token expired"
      description: The request cannot be fulfilled because the Voucher is expired, revoked
   or otherwise malformed. See <a target="blank" href="https://datatracker.ietf.org/doc/html
  /rfc6750.html#section-3">RFC6750</a> and <a target="blank" href="https://datatracker.ietf.
  org/doc/html/rfc9449.html#section-7.1-11">RFC9449</a> for details.

  schemas:
    CredentialClaimsResponse:
      type: object
      properties:
```

```
357          userClaims:
358            description: List of User Claims.
359            type: object
360            properties:
361              given_name:
362                description: Current First Name.
363                type: string
364                example: "Mario"
365              family_name:
366                description: Current Family Name.
367                type: string
368                example: "Rossi"
369              birth_date:
370                description: Date of Birth.
371                type: string
372                example: "1980-01-10"
373              place_of_birth:
374                $ref: "#/components/schemas/place_of_birth"
375              tax_id_code:
376                description: National tax identification number. REQUIRED if personal_admini
    strative_number is absent.
377                type: string
378                example: "TINIT-XXXXXXXXXXXXXXX"
379              personal_administrative_number:
380                description: National unique identifier of a natural person. REQUIRED if tax
    _id_code is absent.
381                type: string
382                example: "XX00000XX"
383          attributeClaims:
384            description: List of Datasets of Attribute.
385            type: array
386            items:
387              type: object
388              properties:
389                object_id:
390                  description: Unique identifier of the Dataset. It MUST NOT contain persona
    l data. Required also if additionalProperties are not present. This parameter MUST be used
     to notify through Signal Hub data updating or availability.
391                  type: string
392                  example: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
393                issuance_date:
394                  description: Administrative validity start date of the Dataset
395                  type: string
396                  example: "2025-01-01"
397                expiry_date:
398                  description: Administrative expiry date of the Dataset.
399                  type: string
```

```yaml
                example: "2025-12-31"
            additionalProperties:
              oneOf:
                - type: string
                - type: integer
                  format: int32
                - type: integer
                  format: int64
                - type: number
                  format: float
                - type: number
                  format: double
                - type: boolean
                - type: array
                  items:
                    $ref: '#/components/schemas/additionalPropertiesArray'
                - type: object
                  additionalProperties:
                    $ref: '#/components/schemas/additionalPropertiesObject'
            required: [object_id]
        metadataClaims:
          description: List of Metadata of Attribute.
          type: array
          items:
            type: object
            properties:
              object_id:
                description: Unique identifier of the Dataset. It MUST NOT contain personal data.
                type: string
                example: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
              description:
                description: Human-Readable description of the Dataset.
                type: string
                example: "Example: Master's Degree in Computer Science"
              status:
                description: |
                  Status of the Dataset. REQUIRED if additionalProperties of attributeClaims are present. Issued and Expired datasets fall within VALID; expiry is verified
                  via metadata claims (e.g. expiry_date, nbf/exp). INVALID indicates active revocation by the AS.
                  For how this status affects the Digital Credential lifecycle managed by the Credential
                  Issuer, see <a target="blank" href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/credential-revocation.html#status-update-by-authentic-sources">Status Update by Authentic Sources</a>.
                type: string
```

```
441                   enum:
442                     - VALID
443                     - INVALID
444                     - SUSPENDED
445                   x-enum-description:
446                     - VALID - Dataset is valid (includes Issued and Expired; expiry checked
    via metadata).
447                     - INVALID - Dataset has been actively revoked by the Authentic Source.
448                     - SUSPENDED - Dataset is temporarily invalid (typically reversible).
449                   example: "VALID"
450                 status_description:
451                   description: Human-Readable description of the Status.
452                   type: string
453                   example: "Example: Master's Degree in Computer Science"
454                 last_updated:
455                   description: REQUIRED if additionalProperties of attributeClaims are prese
    nt. Last time the status or attributes of the Dataset have been updated. Its format is `YY
    YY-MM-DDTHH:MM:SSZ`.
456                   type: string
457                   example: "2025-01-15T10:30:00Z"
458                 interval:
459                   description: Required if userClaims and additionalProperties of AttributeC
    laim parameters are not present. This represents the estimated amount of time (in seconds)
     required before making the request of the attribute claims again.
460                   type: integer
461                   format: int64
462                   example: 864000
463               required: [object_id, description]
464         required: [attributeClaims, metadataClaims]
465       CredentialClaimsRequest:
466         required:
467           - unique_id
468         type: object
469         properties:
470           unique_id:
471             type: string
472             description: ID ANPR or Tax identification number
473           object_id:
474             type: string
475             description: Unique identifier of the Credential dataset. If this parameter is p
    resent only the indicated dataset is returned.
476       additionalPropertiesArray:
477         type: array
478         items:
479           oneOf:
480           - type: string
481           - type: integer
```

```yaml
          format: int32
        - type: integer
          format: int64
        - type: number
          format: float
        - type: number
          format: double
        - type: boolean
        - type: object
          additionalProperties:
            $ref: '#/components/schemas/additionalPropertiesObject'
    additionalPropertiesObject:
      type: object
      oneOf:
        - type: string
        - type: integer
          format: int32
        - type: integer
          format: int64
        - type: number
          format: float
        - type: number
          format: double
        - type: boolean
    place_of_birth:
      description: Place of Birth.
      type: object
      properties:
        country:
          description: "alpha-2 country code as specified in ISO 3166-1"
          type: string
          example: "IT"
        region:
          description: "Name of a state, province, district, or local area"
          type: string
          example: "Roma"
        locality:
          description: "Name of a municipality, city, town, or village"
          type: string
          example: "Roma"
      anyOf:
        - required: [country]
        - required: [region]
        - required: [locality]
    ProblemDetails:
      type: object
      description: RFC7807-compliant problem details object for error responses.
```

```
529       properties:
530         type:
531           type: string
532           format: uri
533           description: An absolute URI that identifies the problem type.
534         title:
535           type: string
536           description: A short, human-readable summary of the problem type.
537         status:
538           type: integer
539           format: int32
540           description: The HTTP status code generated by the origin server for this occurr
    ence of the problem.
541         detail:
542           type: string
543           description: A human-readable explanation specific to this occurrence of the pro
    blem.
544         instance:
545           type: string
546           format: uri
547           description: An absolute URI that identifies the specific occurrence of the prob
    lem.
548       required: [title, status, detail]
```

20.4 Test Plans

The purpose of the test plans is to support implementers, auditors, and conformance test environments in validating the behavior of the Wallet Solutions, Relying Parties and Credential Issuers under various operational and security scenarios.

All test cases are derived from normative rules defined in the above specifications, with no assumptions or extensions.

> **Note**
>
> Please note that the test plans matrix may be subject to future changes.

20.4.1 Structure of the Test Matrix

Each test case is identified by a unique test ID (e.g., WS-001) and categorized using the functional domains defined below. Other category identifiers can be used as well.

- **Discovery**
- **Security**
- **Privacy**
- **Authorization**
- **Authentication**
- **User Experience**
- **Credential Issuance**
- **Credential Presentation**
- **Credential Status**
- **Backup & Restore**

For each test case, the table specifies:

- **Test Case ID**: a unique identifier.
- **Purpose**: the functional area covered by the test.
- **Description**: the requirement being tested, always based on a normative MUST from the specification.
- **Expected Result**: the expected outcome when the solution is implemented correctly.

Signature Evaluation Test Matrix

This section provides the common set of test cases for Wallet Solutions, Relying Parties and Credential Issuers evaluating any signed statements, be these assertions, requests, attestation or Credentials.

Test Case ID	Purpose	Description	Expected Result
ATT-001	Discovery, Security	Evaluation of the issuer	Entities evaluating signed statements establish trust with the issuer and assess its compliance. Undiscoverable Issuers within the federation or unlinkable to any known Trust Anchor, halt any protocol communications.
ATT-002	Discovery, Security	Evaluation of the signature	Entities evaluate signed statements by verifying the signature with the issuer's cryptographic material, provided it is trusted through a well-known Trust Anchor. Any untrusted cryptographic material or invalid signatures halt protocol communications.
ATT-003	Algorithm Verification	Verify that the algorithm specified in the header matches the one used for cryptographic operations.	The algorithm in the header must match the cryptographic operation.
ATT-004	Appropriate Algorithms	Ensure only algorithms listed as MUST or RECOMMENDED in *Cryptographic Algorithms* (page 615) are used. Algorithms listed as MUST NOT (including `none`) are rejected. Cipher suites and hash functions not profiled in that section follow the ACN *Linee guida funzioni crittografiche*. Edwards-curve and post-quantum algorithms are not required by the current profile.	Only approved algorithms are accepted; deprecated or unlisted algorithms are rejected.
ATT-005	Signature Validation	Validate all cryptographic operations and reject if any fail.	All signatures must be valid; any failure results in rejection.

continues on next page

Table 20.23 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
ATT-006	Key Entropy	Cryptographic keys and fresh secrets provide at least 128 bits of security strength as defined in NIST SP 800-57 Part 1. Asymmetric keys used with ES256/ESP256 are P-256 keys generated inside the Keystore or WSCD CSPRNG. Nonces, `jti`, `state` and similar values are CSPRNG output of at least 128 bits and are not sequential.	Keys and secrets meet the entropy minima; weak, undersized, sequential or software-forged keys are rejected. A third party verifies key type and length and, where the platform exposes it, that the key is inside secure hardware (Android `KeyInfo.isInsideSecureHardware` / iOS Secure Enclave token).
ATT-006a	Key Entropy	PKCE `code_verifier`	The `code_verifier` is 43 to 128 unreserved characters as required by **RFC 7636**, generated with a CSPRNG.
ATT-006b	Key Entropy	MRTD and challenge nonces	Challenge identifiers and MRTD PoP nonces have at least 128 bits of entropy, as required for eID Substantial Authentication with MRTD Verification.
ATT-006c	Key Entropy	No imported weak keys	The Wallet Provider rejects Key Attestations for keys that were imported into the Keystore, generated outside the Keystore/WSCD, or whose length does not match the algorithm (for example ES256 with a non-P-256 key).
ATT-007	Issuer Validation	Validate that the cryptographic keys belong to the issuer.	Keys must be verified as belonging to the issuer.
ATT-008	Audience Validation	Validate the audience claim to ensure the token is used by the intended party.	Audience claim must match the intended recipient.
ATT-009	Claim Trust	Do not trust received claims without validation.	Claims must be validated; untrusted claims are rejected.
ATT-010	Explicit Typing	Use explicit typing to prevent COSE/JOSE confusion.	Typing must be explicit and validated.
ATT-011	Cross-JWT Confusion	Prevent COSE/JOSE from being used in unintended contexts.	COSE/JOSE must be contextually validated to prevent misuse.

continues on next page

Table 20.23 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
ATT-012	Substitution Attacks	Ensure COSE/JOSE are not substituted across different contexts.	COSE/JOSE must be validated for context-specific use.
ATT-013	Issued At Validation	Verify that the *issued at* parameter is set to the current time, allowing a grace period not exceeding 120 seconds.	The *issued at* value must be within 120 seconds of the current time.
ATT-014	Expiration Validation	Ensure the *expiration* time is greater than the *issued at* time.	The *expiration* time must be later than the *issued at* time.
ATT-015	Data Model validation	Ensure JOSE/COSE type matches with the defined data model.	The parameters or claims, their values and the schema used to represent them are compliant with the data model.

Trust Evaluation Test Matrix

This section provides the common set of test cases for Wallet Solutions, Relying Parties and Credential Issuers.

Test Case ID	Purpose	Description	Expected Result
ALL-001	Security	Obtaining Trust Anchors public cryptographic materials	Entities obtain the list of Trust Anchors or Certificate Authorities and their public cryptographic key materials, periodically ensuring that these are not expired, revoked or updated. Infrastructure of Trust provides these information through web endpoints and other out of band mechanisms, to facilitate comparison of the provided information to all the entities.

continues on next page

Table 20.24 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
ALL-002	Security	Compliance self evaluation	Entities periodically evaluate their compliance and presence within the federation, checking the trust chain about themselves as still valid, not revoked and compliant with the technical specification. Entities apply the policies, checking that their current configuration is valid with the active policies about them within the federation. Trust chain, evaluated and stored in multiple formats to facilitate interoperability in trust discovery with other entities, are stored by entities and used on occurrence during the data exchange flows. Trust chain about entities are fetched or discovered using the entities' issued assertions.
ALL-003	Discovery	Publication of information about itself	Entities sign and publish all the information about them, containing all the protocol metadata, cryptographic material, trust marks, using the well-known endpoint defined in this specification, making these information publicly discoverable by other entities.
ALL-004	Security	Publication of the historical key registry	Entities sign and publish all the information about the unused or revoked cryptographic material using well known endpoints defined in this specification, making this information publicly discoverable by other entities.

continues on next page

Table 20.24 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
ALL-005	Security	Evaluation of compliance with entities before exchanging data about the User	Entities evaluate trust and compliance with other entities before any information related to a natural or legal person might be exchanged. Bogus configurations don't allow data exchanges.
ALL-006	Security	Evaluation of proof of possession during the use of a signed assertion according to the configured usage ownership confirmation method.	Entities evaluate the confirmation method and apply its protocol to consider valid the signed statement.
ALL-007	Security	Supported cryptography algorithms	Entities evaluate cryptographic usage for compliance with the allowed algorithms.
ALL-008	Security	Replay attacks	Signed statements using unique identifiers are stored until their expiration time and checked against any replay of them.

Federation Authority Test Matrix

This section defines test cases for Federation Authorities (Trust Anchors and Intermediates) responsible for operating the Trust Infrastructure as described in *The Infrastructure of Trust* (page 109). Tests focus on correctness and conformance of:

- Entity Configuration (`.well-known/openid-federation`)
- Subordinate Statements returned by `/fetch`
- Federation registry endpoints (`/list`, `/fetch`, `/trust_mark_status`, `/historical-jwks`)

All validations align with (OID-FED).

Test Case ID	Purpose	Description	Expected Result
FA_001	Discovery, Interoperability	Entity Configuration media type	`GET /.well-known/openid-federation` returns HTTP 200 with `Content-Type: application/entity-statement+jwt`.

continues on next page

Table 20.25 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
FA_002	Security	Entity Configuration signature integrity	The Entity Configuration is a signed JWT; signature verifies using one of the Federation Entity public keys contained in the Entity Configuration `jwks` per OID-FED.
FA_003	Interoperability	Entity Configuration JOSE header parameters	JOSE header contains `alg` (permitted), optional `kid` referencing a key in `jwks`, and `typ` set to `entity-statement+jwt`.
FA_004	Security	Entity Configuration standard claims	Payload includes `iss` and `sub` both equal to the Federation Authority identifier URL; includes `iat` and `exp` as valid Unix timestamps; `exp` > `iat` and current time < `exp`.
FA_005	Security, Interoperability	Entity Configuration common parameters	Payload contains `jwks` and `metadata` with `federation_entity` object including published Federation endpoints as per *Trust Infrastructure and Registry Integration* (page 111).
FA_006	Security	Entity Configuration key material validity	`jwks.keys[*]` entries use permitted algorithms and key sizes; keys are not expired or revoked per `/historical-jwks`; each `kid` is unique.
FA_007	Security	`exp` validation tolerance	Validation rejects Entity Configuration if `exp` is in the past; a maximum clock skew of 120 seconds MAY be applied when comparing current time to `iat`/`exp`.
FA_008	Security	Subordinate Statement signature and lifetime	`GET /fetch?sub={entity}` returns a signed JWT Subordinate Statement; header and payload verify per OID-FED; `iss` equals the issuing Federation Authority; `sub` equals requested entity; `iat`/`exp` valid.

continues on next page

Table 20.25 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
FA_009	Interoperability	Subordinate Statement schema	Subordinate Statement contains the subordinate's public keys (directly or by reference) and applicable metadata policies or metadata, conforming to draft-43 structure.
FA_010	Discovery	Listing subordinates	`GET /list` returns a JSON array or JWT-wrapped list of current subordinate identifiers; response format and media type match the published metadata; HTTP 200.
FA_011	Security	Trust Mark status endpoint	`GET /trust_mark_status?id={tm_id}&sub={entity}` returns current status with HTTP 200 and an integrity-protected object (JWT if advertised). Unknown Trust Mark IDs or subjects return appropriate 4xx.
FA_012	Security	Historical keys endpoint	`GET /historical-jwks` returns revoked/expired keys with revocation reasons. Structure validates as JWKS or JWT-wrapped JWKS per advertised media type.
FA_013	Security	Key rotation propagation	After rotating keys, `/.well-known/openid-federation` and `/historical-jwks` are updated atomically or within a documented maximum propagation window. Verification with the new key succeeds, and the old key appears in historical.
FA_014	Security	Disallow `alg":"none` and weak algorithms	Any Entity Configuration or Subordinate Statement with `alg": "none"` or a disallowed algorithm is rejected; endpoint returns appropriate error (e.g. , 400/422).

continues on next page

Table 20.25 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
FA_015	Security	`kid` resolution and mismatch handling	If JOSE header `kid` is present, it matches a key in the current `jwks`; on mismatch, verification fails with clear error.
FA_016	Interoperability	Endpoint discovery from metadata	`federation_entity` metadata in Entity Configuration includes working URLs for `federation_list_endpoint`, `federation_fetch_endpoint`, `federation_trust_mark_status_endpoint`, and `historical-jwks` (if published); each resolves and responds per its contract.
FA_017	Security	Issuer/subject self-consistency	For Entity Configuration, `iss == sub ==` Authority URL; for Subordinate Statements, `iss` is Authority URL and `sub` is subordinate URL; any deviation is rejected.
FA_018	Interoperability	Media type correctness (fetch/list/status)	`/fetch`, `/list`, `/trust_mark_status`, and `/historical-jwks` return the media types declared in `federation_entity` metadata; JWT-wrapped resources use the correct `Content-Type` (e.g., `application/entity-statement+jwt`, `application/jwk-set+jwt`).
FA_019	Security	Statement replay prevention (`jti` optional)	If `jti` is published, repeated use within the validity window is detected and logged or rejected according to policy; otherwise, uniqueness is not required.

continues on next page

Table 20.25 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
FA_020	Security	Metadata policy application (if used)	When metadata policies are used in Subordinate Statements, the resulting effective metadata computed from policy + source metadata conforms to draft-43 rules; conflicts are resolved deterministically.

Wallet Provider Test Matrix

This section provides the set of test cases for verifying conformance of a Wallet Solution and Wallet Instance implementation to the technical rules defined in the IT-Wallet ecosystem. The test plan is based on the requirements extracted from the following Sections:

- *The Infrastructure of Trust* (page 109)
- *Wallet Solution* (page 269)
- *Digital Credential Issuance* (page 419)
- *Digital Credential Presentation* (page 480)
- *Endpoints* (page 543)
- *Mobile Application Instance* (page 667)

Note

The test cases in this list vary in scope. Some are intentionally written at a high level to confirm the success of complete user flows and the conformance of core architectural and interoperability principles, while others are atomic, targeting single, detailed requirements and verifying specific functional behavior.

Test Cases for Wallet Provider Backend

This section lists the test cases from Sections:

- *Wallet Solution Requirements* (page 269)
- *Wallet Solution Components* (page 276)
- *Wallet Instance* (page 279)
- *Wallet Provider Entity Configuration* (page 305)
- *Wallet Solution Metadata* (page 308)
- e-Service PDND Wallet Provider Catalogue

Test Case ID	Purpose	Description	Expected Result
WP_001	Trust, Security	Entity Configuration publication	An HTTP GET request to the `.well-known/openid-federation` endpoint retrieves the Wallet Provider Entity Configuration.
WP_002	Trust, Security	Entity Configuration cryptography	The Wallet Provider Entity Configuration is a signed JWT that conforms to OID-FED.
WP_002a	Trust, Interoperability	Entity Configuration JWT `alg` header parameter	The Entity Configuration JWT header contains an `alg` set to a permitted signature algorithm (e.g., `ES256`) and not `none`.
WP_002b	Trust, Interoperability	Entity Configuration JWT key identifier header parameter	The Entity Configuration JWT header contains a `kid` equal to the thumbprint of the public key used to sign the Entity Configuration.
WP_002c	Trust, Interoperability	Entity Configuration JWT type header parameter	The Entity Configuration JWT header contains `typ` set to `entity-statement+jwt`.
WP_002d	Trust, Interoperability	Entity Configuration issuer and subject payload claims	The Entity Configuration JWT payload contains `iss` and `sub`, both set to the public URL of the Wallet Provider.
WP_002e	Trust, Interoperability	Entity Configuration validity	The Entity Configuration JWT payload contains `iat` and `exp` as valid Unix timestamps.
WP_002f	Trust, Interoperability	Entity Configuration authority hints	The Entity Configuration JWT payload contains `authority_hints` as an array of valid URLs of immediate superior entities.

continues on next page

Table 20.26 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_002g	Trust, Interoperability	Entity Configuration signing keys	The Entity Configuration JWT payload contains `jwks` with a valid JWKS of the Wallet Provider's Federation Entity public signing keys.
WP_002h	Trust, Interoperability	Entity Configuration metadata	The Entity Configuration JWT payload contains a `metadata` object that includes `openid_wallet_provider` metadata and optionally `federation_entity`, each following its schema.
WP_003	Trust, Interoperability	Metadata key usage	The public keys in the metadata JSON Object are designated exclusively for signing and/or encryption when the entity acts as a Wallet Provider (e.g., for Wallet Instance Attestations (WIA)).
WP_004	Trust, Interoperability	Metadata key reference	To reference public keys, the metadata JSON Object contains exactly one of the following claims: `jwks`, `jwks_uri`, or `signed_jwks_uri`.
WP_004a	Trust, Interoperability	JWKS by value	If `jwks` is present, it contains a valid JWKS document passed by value.
WP_004b	Trust, Interoperability	JWKS by reference	If `jwks_uri` is present, the value is an HTTPS URL to a JWKS that resolves and validates.
WP_004c	Trust, Interoperability	Signed JWKS URI	If `signed_jwks_uri` is present, the value is an HTTPS URL to a JWT whose payload is a JWKS and is signed with a Federation Entity Key. The `signed_jwks_uri` resource is served with content type `application/jwk-set+jwt`. A successful `signed_jwks_uri` retrieval returns `HTTP 200`.
WP_005	Trust, UX	Portal 2FA enforcement	The Wallet Provider portal correctly enforces two-factor authentication (`2FA`) before allowing Users to perform any operation.
WP_006	Wallet Revocation, Security	Portal Wallet Instances status view	The Wallet Provider portal correctly allows Users to view the current status of all Wallet Instances linked to their account.
WP_007	Wallet Revocation, Security	User-initiated revocation trigger	Wallet Provider implements and supports revocation requests initiated by Users via its portal.
WP_007a	Wallet Revocation, Security	Portal instance revocation	The Wallet Provider portal correctly allows Users to revoke a specific Wallet Instance or all Wallet Instances.

continues on next page

Table 20.26 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_008	Wallet Revocation, Security	PID Provider-initiated revocation trigger	Wallet Provider implements and supports revocation requests triggered by PID Providers or IT-Wallet ID Providers via the PDND Endpoint.
WP_009	Wallet Revocation, Security	Legal Authorities-initiated revocation trigger	Wallet Provider implements and supports revocation requests triggered by Legal Authorities or Supervisory Bodies (e.g., illegal activities).
WP_010	Trust, Interoperability	Wallet Instance revocation mechanism	When Wallet Provider triggers the revocation for a specific Wallet Instance at any time, that instance is terminated and can no longer perform any functions.
WP_011	Trust, Security	Wallet Instance integrity verification	Wallet Provider periodically evaluates Wallet Instance's integrity, authenticity, and genuineness using the Play Integrity API (Android) or App Attest (iOS).
WP_012	Lifecycle, Interoperability	Backend component architecture	The Wallet Provider Backend supports all the components (Frontend, API Interface, Wallet Instance Lifecycle Management, and Trust & Security) as shown in *Figure of Wallet Solution High Level Architecture* (page 270).

Test Cases for Wallet Instance

This section lists the test cases from Sections:

- Configuration of the Federation
- Trust Evaluation Mechanism
- *Wallet Solution Requirements* (page 269)
- *Wallet Solution Components* (page 276)
- *Wallet Instance* (page 279)
- Error Handling for Wallet Instance Management
- Mobile Application Instance Initialization

Test Case ID	Purpose	Description	Expected Result
WP_013	Lifecycle, Interoperability	Frontend component architecture	Wallet Instance supports all the components (User Interface, Lifecycle Management, Issuer, Presentation, Backup/Restore, and Secure Storage) as shown in *Figure of Wallet Solution High Level Architecture* (page 270).
WP_014	Trust, Security	Keystore / WSCD profile	The current implementation profile uses only a Local Internal hardware-backed Keystore (Android TEE or StrongBox, iOS Secure Enclave). The Wallet Instance generates and stores User private keys in that Keystore and does not use a software-only keystore. WSCD denotes a High-certifiable device (HSM or smart card, at least Common Criteria EAL4+ AVA_VAN. 5).
WP_014a	Trust, Security	Hardware-backed Keystore only	Attempts to generate or import User private keys in a software-only keystore are rejected. Cryptographic operations for WIA, KA, IT-Wallet ID and (Q)EAA key binding use the Local Internal Keystore.
WP_014b	Trust, Security	Non-exportability of private keys	The Keystore or WSCD does not allow export of User private keys in clear. If a device reports the private key as exportable, the Wallet Provider rejects the Key Attestation and does not activate the Wallet Instance.
WP_014c	Trust, Security	Sole Control and User authentication	The Keystore or WSCD requires User authentication (wallet unlock: PIN or biometric) before any signature or key-use operation. The User retains exclusive control of the private keys.
WP_014d	Trust, Security	PID High vs IT-Wallet ID key storage	Keys of IT-Wallet ID and (Q)EAA MAY be stored in the Local Internal Keystore (eIDAS Substantial). Keys of a PID at LoA High MUST be stored only in a WSCD. The current profile does not issue a PID High; a TEE Protection Profile or similar report is not accepted as evidence of LoA High.
WP_014e	Trust, Security	WIA keys collocation	Private keys bound to the WIA are generated and stored in the same Keystore or WSCD that the WIA attests.
WP_014f	Trust, Security	Mobile Relying Party Instance keys	A Mobile Relying Party Instance that does not store User identity keys MAY use the Local Internal Keystore. It is not required to use a High-certifiable WSCD.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_015	Lifecycle, UX	Android/iOS compatibility	Wallet Instance installs and operates with full functionality on both Android and iOS devices and is available on the Play Store and App Store.
WP_016	Trust, Issuance, Interoperability	Retrieve and maintain Trust Anchor metadata	Wallet Instance fetches and refreshes the Trust Anchor's Entity Configuration before use and keeps it up-to-date.
WP_017	Trust, Issuance, Interoperability	Validate Trust Anchor authenticity	Wallet Instance compares out-of-band public keys to those in the Trust Anchor's Entity Configuration and discards non-matching keys, logs discrepancies, and uses only matches.
WP_018	Lifecycle, Trust, Security	Periodic trust reestablishment	Wallet Instance periodically and successfully obtains a fresh Wallet Instance Attestation (WIA) from its Wallet Provider.
WP_019	Trust, Security	Wallet Instance Attestation (WIA) content	The Wallet Instance Attestation (WIA) contains all required claims and data points that attest to the device's integrity and security status.
WP_019a	Trust, Security	No attestation for unverified Wallet Instances	Wallet Provider rejects an attestation request from a Wallet Instance that fails authenticity, integrity, or genuineness checks.
WP_019b	Trust, Security	Attestation-ephemeral key binding	The Wallet Instance Attestation (WIA) contains a cryptographic binding to Wallet Instance's ephemeral public key that is successfully verified.
WP_020	Trust, Security	Wallet Instance Attestation (WIA) signing	The Wallet Instance Attestation (WIA) is signed by its authorized Wallet Provider, as officially listed by the overseeing Registration Authority.
WP_021	Wallet Initialization / Registration, Lifecycle, Security	Device security verification	Wallet Instance prevents activation on a device that does not meet the Wallet Provider's defined minimum security requirements.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_022	Wallet Initialization / Registration, Lifecycle, Security	Key Attestation from Local Internal Keystore	Wallet Instance checks and confirms the availability of the platform Key Attestation APIs of the Local Internal Keystore (StrongBox/TEE or Secure Enclave) and obtains a Key Attestation (KA) for keys generated in that Keystore.
WP_022a	Wallet Initialization / Registration, Lifecycle, Security	KA `key_storage` and `user_authentication`	The KA JWT sets `key_storage` and `user_authentication` to ISO 18045 values that match the attack-potential resistance of the actual Keystore or WSCD and of the User authentication method. The Wallet Provider documents the mapping used for TEE, StrongBox and Secure Enclave. Values MUST NOT claim `iso_18045_high` for a Local Internal Keystore.
WP_022b	Wallet Initialization / Registration, Lifecycle, Security	KA for remote or local-external WSCD	If a future profile uses a remote WSCD (HSM) or a local-external WSCD (smart card, token), the KA is produced by that WSCD, not by the phone TEE. Sole Control of the User over remote keys is verified. This case is out of the current profile.
WP_023	Wallet Initialization / Registration, Wallet Instance Attestation (WIA) Issuance, Lifecycle, Trust	Wallet Provider federation discovery	Wallet Instance successfully uses Federation API endpoints (`.well-known/openid-federation`, `/fetch`) to retrieve current metadata and configurations of the Wallet Provider.
WP_024	Wallet Initialization / Activation, Lifecycle, Security	User consent	During the Wallet initialization, Wallet Provider explicitly obtains the User's consent before creating a User account.
WP_025	Wallet Activation, Lifecycle, UX	Wallet unlock method setup	Wallet Instance prompts the User to set their preferred unlock method (PIN or biometric), and then successfully configures the chosen method.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_026	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Ephemeral key pair for attestation	To perform a Wallet Instance Attestation (WIA) request, Wallet Instance successfully generates a new ephemeral asymmetric key pair.
WP_027	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Device security flaw verification	Wallet Provider verifies the device meets its minimum security requirements and is free of known security flaws; if not, the Wallet Instance Attestation (WIA) Request is rejected.
WP_028	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Time-limited Wallet Instance Attestation (WIA)	The issued WIA meets *the Wallet Instance Attestation lifetime* (page 272). A formulation limited to «short-lived» is not sufficient.
WP_029	Wallet Instance Attestation (WIA) Issuance, Data Model and Lifecycle, Interoperability	HTTP 200 / JSON response envelope	Upon successful validation of the Wallet Instance Attestation (WIA) Issuance Request, the Wallet Provider returns 200 OK with Content-Type: application/json, containing the Wallet Instance Attestation (WIA) as its structure defined in Wallet Instance Attestation (WIA) JWT.
WP_029a	Wallet Instance Attestation (WIA) Issuance, Data Model and Lifecycle, Security	Wallet Instance Attestation (WIA) format	Wallet Provider provides the Wallet Instance Attestation (WIA) in JWT format signed by the Wallet Provider, and confirming the structures defined in Wallet Instance Attestation (WIA) JWT.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_029b	Wallet Instance Attestation (WIA) Issuance, Data Model and Lifecycle, Security	No PII in Wallet Instance Attestation (WIA)	The Wallet Instance Attestation (WIA) payload contains no personally identifiable information (PII) about the User.
WP_030	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Wallet Instance Attestation (WIA) integrity verification	Wallet Instance verifies the signature of the received Wallet Instance Attestation (WIA).
WP_031	Wallet Instance Attestation (WIA) Issuance, Trust, Security	Wallet Instance Attestation (WIA) trust verification	Wallet Instance verifies and confirms that the Wallet Instance Attestation (WIA) issuer (i. e., Wallet Provider) is a trusted member of the Federation, before accepting the Wallet Instance Attestation (WIA).
WP_032	Wallet Revocation, Lifecycle, Security	User-initiated revocation	The User successfully initiates a revocation either through the Wallet Instance built-in revocation function or via an external user agent.
WP_032a	Wallet Revocation, Lifecycle, Interoperability	User login to portal with 2FA	To perform a Wallet Instance revocation, the User successfully logs into the Wallet Provider web portal with `2FA`.
WP_032b	Wallet Revocation, Lifecycle, UX	User selection of a Wallet Instance to revoke	The User views the status of all their linked Wallet Instances, and chooses one to be revoked.
WP_033	Wallet Revocation, Lifecycle, Interoperability	Wallet Instance Revocation Request	Wallet Instance requests revocation of a specified instance by sending a Wallet Instance Revocation Request to the Wallet Instance Management Endpoint.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_034	Wallet Revocation, Lifecycle, Security	Timely revocation alerts	Wallet Provider informs the User within 24 hours of a Wallet Instance revocation.
WP_034a	Wallet Revocation, Lifecycle, Security	Out-of-band revocation alerts	Wallet Provider notifies the User of a Wallet Instance revocation through an out-of-band channel (e.g., email or SMS).
WP_034b	Wallet Revocation, Lifecycle, Security	Clear revocation notification with re-activation guidance	The revocation notification clearly explains the reason for revocation and provides actionable steps for re-activation, if applicable.
WP_035	Lifecycle, Security	Error handling for Wallet Instance management	When any error occurs during Wallet Instance Registration/Initialization flow, or during Status Retrieval and Revocation requests, Wallet Provider returns an error response to Wallet Instance that conforms to the standards defined in **RFC 7231** (for HTTP status code) and **RFC 7807** (for problem details).
WP_035a	Lifecycle, Interoperability	Standard error response format	The error response returned by the Wallet Provider includes a `Content-Type` header of `application/json` and a JSON body containing the mandatory `error` and `error_description` parameters.
WP_036	Lifecycle, Interoperability	Handling malformed requests (400)	When a request is malformed or missing required parameters, Wallet Provider returns a response with the HTTP status code (`400 Bad Request`) and with the error code `bad_request`.
WP_037	Lifecycle, Interoperability	Semantic validation error (422)	When a request is well-formed but fails semantic validation, Wallet Provider returns an optional response with the HTTP status code (`422 Unprocessable Content`) and with the error code `validation_error`.
WP_038	Lifecycle, Interoperability	Internal server error (500)	In the event of an unhandled internal failure, Wallet Provider returns a response with the HTTP status code (`500 Internal Server Error`) and with the error code `server_error`.

continues on next page

Table 20.27 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_039	Lifecycle, Interoperability	Service unavailable (503)	When the service is temporarily offline or unable to handle requests, Wallet Provider returns a response with the HTTP status code (`503 Service Unavailable`) and with the error code `temporarily_unavailable`.
WP_040	Wallet Initialization / Registration, Lifecycle, Interoperability	Initialization failure (device integrity)	When a Wallet Instance Initialization request comes from a device that does not meet the provider's security requirements, the provider returns a `403 Forbidden` response with the error code `integrity_check_error`.
WP_041	Wallet Revocation, Lifecycle, Interoperability	Unauthorized retrieval request (401)	When a Wallet Instance status retrieval request is made without valid authentication credentials, Wallet Provider returns a response with the HTTP status code (`401 Unauthorized`) and with the error code `unauthorized`.
WP_042	Wallet Revocation, Lifecycle, Interoperability	Forbidden retrieval request (403)	When a User attempts to retrieve a Wallet Instance status they do not have permission for, Wallet Provider returns a response with the HTTP status code (`403 Forbidden`) and with the error code `forbidden`.
WP_043	Wallet Revocation, Lifecycle, Interoperability	Unauthorized revocation request (401)	When a Wallet Instance Revocation request is made without valid authentication credentials, Wallet Provider returns a response with the HTTP status code (`401 Unauthorized`) with the error code `unauthorized`.
WP_044	Wallet Revocation, Lifecycle, Interoperability	Forbidden revocation request (403)	When a User attempts to revoke a Wallet Instance they do not have permission for, Wallet Provider returns a response with the HTTP status code (`403 Forbidden`) and with the error code `invalid_request`.

Test Cases for Issuance Phase

This section lists the test cases from Sections:

- *Digital Credential Issuance* (page 419)
- Trust Evaluation Mechanism
- *Credential Issuer Endpoints* (page 562)

Test Case ID	Purpose	Description	Expected Result
WP_045	Issuance, Interoperability	Credential Issuer Discovery	Wallet Instance successfully discovers trusted Digital Credential Issuers using the Credential Catalogue.
WP_045a	Trust, Issuance, Interoperability	Fetch the Digital Credential catalogue	Wallet Instance successfully sends an HTTP GET request to the Digital Credentials Catalogue Endpoint, using the `application/jose` media type.
WP_046	Issuance, Interoperability	Discover Credential Issuer dynamically from federation metadata	Wallet Instance successfully uses Federation API endpoints (`.well-known/openid-federation`, `/fetch`) to retrieve current metadata and configurations of the Credential Issuer.
WP_046a	Trust, Issuance, Security	Validate Credential Issuer's full Trust Chain to the Trust Anchor	Wallet Instance builds and verifies the full Trust Chain from the Credential Issuer through Intermediaries to the root Trust Anchor, ensuring each signature is valid and confirms the root.
WP_047	Issuance, Interoperability	Third-Party-Initiated flow: Credential Offer QR scan & decode	In a Third-Party-Initiated cross-device flow, Wallet Instance successfully scans the QR code, and retrieves the `credential_offer` or the `credential_offer_uri` parameter.
WP_048	Issuance, Interoperability	Third-Party-Initiated flow: Credential Offer parsing	In a Third-Party-Initiated flow, Wallet Instance extracts and validates all required parameters present in the Credential Offer.
WP_049	Issuance, Interoperability	Issuer-Initiated flow: Authorization Server identifier check	Wallet Instance reads the Credential Issuer's metadata `authorization_servers` and finds the configured Authorization Server identifier in `authorization_servers`.
WP_050	Issuance, Interoperability	Issuer-Initiated flow: Digital Credential IDs validation	Wallet Instance verifies whether each Digital Credential identifier in the `credential_configuration_ids` is supported by the Credential Issuer; any unsupported causes rejection.
WP_050a	Trust, Issuance, Interoperability	Policy & Trust Marks enforcement	Wallet Instance verifies that the Credential Issuer is authorized to issue the requested Digital Credential through the application of metadata policies and the verification of Trust Marks.

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_050b	Trust, Issuance, Interoperability	Supported ID confirmation	Wallet Instance verifies that every requested `credential_configuration_id` appears in the Credential Issuer's `credential_configurations_supported` metadata; missing IDs cause the offer to fail.
WP_051	Issuance, Interoperability	Credential Request using OAuth 2.0 code flow	Wallet Instance successfully requests PID, IT-Wallet ID or (Q)EAA from the corresponding Provider using the Authorization Code Flow per OpenID4VCI.
WP_052	Issuance, Interoperability	PAR Request: payload construction	Wallet Instance generates a fresh PKCE `code_verifier`, Wallet Instance Attestation (WIA) PoP JWT, and `state` value, then wraps them into a Request Object signed with its private key per **RFC 9126**, and posts it to the PAR Endpoint.
WP_052a	Issuance, Interoperability	Secure PKCE `code_verifier` generation	Wallet Instance creates the `code_verifier` following **RFC 7636** recommendations, as a high-entropy random string using unreserved characters with a length between 43 and 128.
WP_052b	Issuance, Interoperability	Wallet Instance Attestation (WIA) PoP JWT creation	Wallet Instance generates the Wallet Instance Attestation (WIA) PoP JWT (per OAuth-Client-Attestation-PoP parameters as defined in OAUTH-ATTESTATION-CLIENT-AUTH) and binds it to the same ephemeral public key referenced in the Wallet Instance Attestation (WIA)'s `cnf.jwk`.
WP_052c	Issuance, Security	Wallet Instance Attestation (WIA) PoP JWT signing	Wallet Instance signs the PoP JWT with the ephemeral private key corresponding to the public key in the Wallet Instance Attestation (WIA)'s `cnf.jwk`.
WP_052d	Issuance, Interoperability	Specify Digital Credential types	Wallet Instance embeds correct Digital Credential types in the Request Object using both the `scope` parameter (per [OPENID4VC-HAIP]) and the `authorization_details` parameter (per RAR **RFC 9396**).
WP_053	Issuance, Security	Authorization Request	Wallet Instance sends an Authorization Request to the Credential Issuer Authorization Endpoint using the received `request_uri` and `client_id`.
WP_053a	Issuance, Security	Enforce one-time `request_uri`	Wallet Instance submits the Authorization Request with a freshly obtained `request_uri`, and that `request_uri` is accepted only once.

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_054	Issuance, Interoperability	Validate Authorization Response: required parameter	Wallet Instance checks that the Authorization Response contains all required parameters (`code`, `state`, and `iss`); missing parameters trigger error handling.
WP_054a	Issuance, Security	Validate Authorization Response: verify state matching	Wallet Instance compares the returned `state` value with the one originally sent in the Request Object (**RFC 6749**) and continues only if they match exactly.
WP_054b	Issuance, Security	Validate Authorization Response: confirm Credential Issuer identity	Wallet Instance checks that the `iss` parameter in the Authorization Response matches the identifier of the Credential Issuer with which it initiated the communication (**RFC 9207**).
WP_055	Issuance, Interoperability	Send a Token Request	Wallet Instance sends a POST request to the Token Endpoint with a URL-encoded body, including the received `code` (in the Authorization Response), the same `redirect_uri` (in the Request Object), and the `code_verifier`.
WP_055a	Issuance, Interoperability	Token request proof parameters	The Token Request carries the required proofs in headers: a DPoP proof JWT, a Wallet Instance Attestation (WIA) JWT, and Wallet Instance's PoP JWT (per OAuth-Client-Attestation and OAuth-Client-Attestation-PoP as defined in OAUTH-ATTESTATION-CLIENT-AUTH).
WP_055b	Issuance, Interoperability	Generate DPoP key pair/proof	Wallet Instance generates a fresh DPoP key pair and its DPoP proof JWT following the instructions provided in **RFC 9449#section-4** for the Token Request to the Credential Issuer.
WP_055c	Issuance, Security	Sign DPoP proof	Wallet Instance signs the DPoP proof JWT with the DPoP private key generated for this scope.
WP_055d	Issuance, Interoperability	Token request proof parameters	The Wallet Instance Attestation (WIA) JWT is signed using the private key bound to Wallet Instance, where its related public key is provided within the Wallet Instance Attestation (WIA) (`cnf.jwk` claim).

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_056	Issuance, Interoperability	Credential Request	Wallet Instance sends a POST request to the Credential Endpoint including the Access Token, DPoP Proof JWT, Credential type, and the valid PoP of Credential-bound key.
WP_056a	Issuance, Security	Fetch nonce for the Digital Credential key proof	Wallet Instance sends a POST request to the Credential Issuer's Nonce Endpoint and obtains a fresh nonce, which is then used to generate the PoP of the key material bound to the issued Digital Credential.
WP_056b	Issuance, Security	Fresh DPoP proof	Wallet Instance generates a fresh DPoP proof with the same key used for the Token Request DPoP proof, and in accordance with **RFC 9449#section-4**.
WP_056c	Issuance, Security	Match the Credential key proof to the DPoP key	Wallet Instance includes a `proofs` object of type `jwt` in the Credential Request that demonstrates possession of the cryptographic key material; the `jwk` value in this object matches the public key referenced in the DPoP proof.
WP_057	Issuance, Interoperability	Multiple Digital Credential request	When a Credential Issuer offers multiple Digital Credentials, Wallet Instance makes a separate and correctly formatted Credential Request for each Digital Credential it intends to accept.
WP_058	Issuance, Interoperability	Batch Credential Request	In case of Batch Issuance, Wallet Instance sends a batch Credential request to the Credential Endpoint that includes: the Access Token, a DPoP proof JWT, the credential type, and a `proofs` parameter containing all key proofs.
WP_058a	Issuance, Security	Batch Issuance: key generation	In case of Batch Issuance, Wallet Instance generates a number of fresh Digital Credential key pairs equal to the `batch_size` value.
WP_058b	Issuance, Interoperability	Batch Issuance: PoP of batch Credential keys	In case of Batch Issuance, Wallet Instance generates N key proofs using the provided `c_nonce`, one for each Digital Credential in the batch, where N equals the `batch_size` value.
WP_059	Issuance, Interoperability	Validate Credential Response for required parameters	Wallet Instance inspects the Credential Response payload, verifying all mandatory PID, IT-Wallet ID or (Q)EAA parameters are present and valid as defined in *Table of Credential Response* (page 586); if any parameter is missing or invalid, it rejects the response with an error.

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_060	Issuance, Interoperability	Verify Digital Credential type/schema	Wallet Instance retrieves the issued Digital Credential from the response's `credential` claim, verifies that its type matches the requested type, and validates the schema against *Digital Credential Data Model* (page 357); if either check fails, it rejects the Digital Credential.
WP_061	Issuance, Security	Validate Credential Issuer Trust Chain	Wallet Instance verifies the Digital Credential's Trust Chain from its header to confirm that the Credential Issuer is trusted.
WP_062	Issuance, Interoperability	Format-specific Digital Credential verification	Wallet Instance verifies whether a Digital Credential is in SD-JWT VC or mdoc-CBOR format, and then runs the appropriate verification. If it fails, it rejects the Credential.
WP_062a	Issuance, Security, Interoperability	Verify SD-JWT Credential integrity	Wallet Instance obtains the `alg` and `kid` values from the SD-JWT header, retrieves the corresponding public key, and verifies the signature; if invalid, the SD-JWT Credential is rejected.
WP_062b	Issuance, Security, Interoperability	Verify mdoc-CBOR Credential integrity	Wallet Instance extracts the signature algorithm from the protected header of the mdoc-CBOR `COSE_Sign1`, retrieves the Credential Issuer's public key from the `kid` or `x5chain` in the unprotected header, and verifies the COSE signature over the MSO; if invalid, the Credential is rejected.
WP_063	Issuance, Privacy	User consent for credential storage	Wallet Instance prompts the User for consent and stores the Digital Credential only after explicit approval.
WP_064	Issuance, Interoperability	Notification Handling	Wallet Instance sends an HTTP POST request to the Notification Endpoint with `Content-Type: application/json`.
WP_064a	Issuance, Interoperability	Notification Parameters	The Notification Request payload contains the `notification_id` from the Credential Response and an `event` value of either `credential_accepted`, `credential_deleted`, or `credential_failure`; it may also include a valid ASCII `event_description`.
WP_064b	Issuance, Privacy	Protect User privacy in notification content	The `event_description` contains only generic, user-neutral text and does not disclose User behavior or device status.

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_065	Issuance, Security	Handle deferred issuance	Wallet Instance evaluates the Credential Response; if it contains both `transaction_id` and `interval`, Wallet Instance recognizes the flow as deferred issuance.
WP_066	Issuance, Interoperability	Deferred issuance request after interval	Wallet Instance submits a Deferred Credential Request only after the required `interval` has passed.
WP_066a	Issuance, Interoperability	Deferred issuance request with transaction_id	Wallet Instance sends a Deferred Credential Request as an HTTP POST with `Content-Type: application/json`, and the request body contains the required `transaction_id`.
WP_066b	Issuance, Interoperability	Deferred request with still-valid Access Token	Wallet Instance includes the existing Access Token in the deferred request if the `interval` parameter value is less than the expiration time set for the Access Token.
WP_066c	Issuance, Interoperability	Fresh DPoP-bound Access Token via the Refresh	When the existing Access Token would expire before the deferred request can be made, Wallet Instance obtains a new DPoP-bound Access Token via the Refresh Token flow.
WP_067	Issuance, Interoperability	New Credential Issuance flow	When Wallet Instance fails to refresh an expiring Access Token, it initiates an entirely new Credential Issuance flow.
WP_068	Issuance, Interoperability	Refresh Token Request	Wallet Instance sends a POST request to the Credential Issuer's Token Endpoint with `grant_type=refresh_token`, a valid refresh_token, a DPoP header with a fresh DPoP proof JWT, and OAuth headers carrying the Wallet Instance Attestation (WIA) JWT and its PoP.
WP_068a	Issuance, Security	Refresh Token: generate proofs	For a refresh Access Token request, Wallet Instance generates a new DPoP JWT and a new Wallet Instance Attestation (WIA) PoP, and includes them in the request.
WP_068b	Issuance, Interoperability	Refresh Token: maintain binding	Wallet Instance reuses the same key bound to the Wallet Instance Attestation (WIA) PoP and the DPoP proof from the original Access Token request, ensuring the refreshed Access Token remains cryptographically bound and valid.
WP_069	Issuance, Security	Check Digital Credential status	Wallet Instance verifies the status of each stored Digital Credential by retrieving and validating either a Status List Token (per *Status List Token* (page 413)).

continues on next page

Table 20.28 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_070	Issuance, Security	Re-issuance flow: detect re-issuance necessity (update status)	Wallet Instance updates a Digital Credential when the Status List shows `0x03` (`UPDATE`) or `0x0F` (`ATTRIBUTE_UPDATE`) for that Credential.
WP_071	Issuance, Security	Re-issuance flow: verify Access Token validity for re-issuance	When an update is detected for a Digital Credential, Wallet Instance verifies the validity of the associated Access Token. If the token is valid, Wallet Instance proceeds with Credential re-issuance.
WP_071a	Issuance, Security	Re-issuance flow: refresh expired Access Token	If the Access Token is expired but a valid Refresh Token is available, Wallet Instance initiates a Refresh Token flow to obtain a new DPoP-bound Access Token, following the Refresh Token Flow.
WP_071b	Issuance, Security	Restart issuance on full expiry	Wallet Instance re-authenticates the User when initiating a new Credential Issuance flow.
WP_072	Issuance, Security	Retrieve refreshed Credential	Wallet Instance, with a valid DPoP-bound Access Token, sends a request to the Credential Endpoint and successfully retrieves the updated Digital Credential.
WP_073	Issuance, Security and Privacy	Delete old Digital Credentials	After storing a new Digital Credential, Wallet Instance deletes the previous version so that only the latest Digital Credential remains stored.
WP_073a	Issuance, Security and Privacy	Delete old batch Credentials	When the Wallet Instance receives and stores a new batch of the same Credential with the same claims, it deletes the previous Credentials.
WP_074	Issuance, UX	Consent on new Digital Credential store with `user_attribute` update	If the Digital Credential update involves changes to the User's attribute values (`attribute_update`), Wallet Instance prompts the User for consent before storing the new refreshed Digital Credential.
WP_075	Issuance, UX	No consent on new Digital Credential store without `user_attribute` update	If the Digital Credential update only concerns credential metadata changes (`update`), Wallet Instance automatically stores the new refreshed Digital Credential without requesting additional consent from the User.

Test Cases for Presentation Phase

This section lists the test cases from Section *Digital Credential Presentation* (page 480), covering both the **Remote Flow** and the **Proximity Flow** presentation phases.

Test Case ID	Purpose	Description	Expected Result
WP_076	Remote-flow, Presentation, Interoperability	Obtain Authorization Request URL in Same Device flow	In the Same Device flow, Wallet Instance successfully obtains a URL and extracts the following parameters: `client_id`, `request_uri`, `state`, and `request_uri_method`.
WP_077	Remote-flow, Presentation, Interoperability	Obtain Authorization Request URL in Cross Device flow	In the Cross Device flow, Wallet Instance successfully scans and parses a QR Code to extract the following parameters: `client_id`, `request_uri`, `state`, and `request_uri_method`.
WP_078	Remote-flow, Trust, Presentation, Interoperability	Relying Party identity discovery	Wallet Instance successfully uses Federation API endpoints (`.well-known/openid-federation`, `/fetch`) to retrieve current metadata and configurations of the Relying Party.
WP_079	Remote-flow, Trust, Presentation, Interoperability	Relying Party Trust Chain evaluation	Wallet Instance successfully validates the Trust Chain of the Relying Party (provided statically or built through a Federation Entity Discovery process) from the Trust Anchor down to the Relying Party itself, confirming that the Relying Party is a recognized and trusted member of the federation.
WP_080	Remote-flow, Trust, Presentation, Interoperability	Relying Party Trust Mark check	Wallet Instance successfully evaluates any Trust Marks included in the Relying Party's Entity Configuration to ensure they are valid and indicate compliance with federation policies.
WP_081	Remote-flow, Trust, Presentation, Security	Validate `request_uri`	Wallet Instance checks the `request_uri` against its locally cached list of allowed URIs (from the `openid_credential_verifier.request_uris` in the Trust Chain) and only proceeds when it finds an exact match; if no match is found, it rejects the request.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_082	Remote-flow, Presentation, Interoperability	GET Request Object	If the `request_uri_method` in the initial request is `get` or is absent, Wallet Instance sends an HTTP GET request to the Relying Party's `request_uri` endpoint, and successfully retrieves a signed Request Object JWT.
WP_083	Remote-flow, Presentation, Interoperability	POST Request Object	If the `request_uri_method` in the initial request is `post`, Wallet Instance sends an HTTP POST request to the Relying Party's `request_uri` endpoint, including its metadata as `wallet_metadata` and `wallet_nonce` parameters, with content type `application/x-www-form-urlencoded`, and successfully retrieves a signed Request Object JWT.
WP_083a	Remote-flow, Presentation, Interoperability	Construct `wallet_metadata`	Wallet Instance formats `wallet_metadata` as a JSON object per *Request URI Request* (page 495) that includes `vp_formats_supported`, `client_id_prefixes_supported`, `authorization_endpoint`, `response_types_supported` set to `["vp_token"]` when present.
WP_083b	Remote-flow, Presentation, Privacy	Exclude PII in `wallet_metadata`	The `wallet_metadata` JSON contains no User-identifiable or device-specific fields (e.g. names, hardware details).
WP_083c	Remote-flow, Presentation, Security	Generate replay nonce	A fresh `wallet_nonce` is generated and included in the POST request payload alongside the `wallet_metadata` to mitigate replay attacks.
WP_084	Remote-flow, Trust, Presentation, Security	Relying Party public key retrieval	Wallet Instance fetches the correct Relying Party's public key exclusively from the `metadata.openid_credential_verifier.jwks` field located within the Relying Party's Entity Configuration provided by the Trust Chain, and using the key identifier (`kid`) from the request JWT header.
WP_085	Remote-flow, Presentation, Security	Request Object signature verification	Wallet Instance confirms the integrity of the signed Request Object by successfully performing the cryptographic signature validation against the Relying Party's public key.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_086	Remote-flow, Presentation, Security	Match `client_id` across contexts	Wallet Instance confirms that the `iss` claim in the signed Request Object exactly equals both the `client_id` it originally extracted from the Authorization Request URL and the `sub` value in the Relying Party's Entity Configuration. Any discrepancy causes the request to be rejected.
WP_087	Remote-flow, Presentation, Security	Check Relying Party eligibility	Wallet Instance authorizes the request and proceeds with the flow when the Relying Party's metadata, policies, and a valid Trust Mark collectively confirm its permission to request the specified credential.
WP_088	Remote-flow, Presentation, Privacy	User consent for disclosure	Wallet Instance requests the User's consent by presenting a UI screen that clearly displays the verified identity of the Relying Party and a list of all requested data attributes.
WP_089	Remote-flow, Presentation, UX	Handle `request_uri` endpoint errors	When the Relying Party's `request_uri` endpoint responds with an error, Wallet Instance displays a clear, human-readable message to the User explaining the failure and securely terminates the flow.
WP_089a	Remote-flow, Presentation, UX	Logging `request_uri` errors	Wallet Instance internally logs the detailed error information.
WP_089b	Remote-flow, Presentation, UX	Recovering from `request_uri` errors	For a recoverable error, such as `server_error`, Wallet Instance prompts the User with a specific recovery action, such as suggesting a retry or "Scan QR again" option, if applicable.
WP_090	Remote-flow, Presentation, Interoperability	Notify Relying Party on invalid request	If the Request Object is invalid or fails verification, Wallet Instance sends an Authorization Error Response via HTTP POST to the Relying Party's `response_uri` Endpoint.
WP_091	Remote-flow, Presentation, Interoperability	Send Authorization Response	If the Request Object verification is valid, Wallet Instance sends an Authorization Response via HTTP POST to the Relying Party's `response_uri` as per `response_mode=direct_post.jwt`.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_091a	Remote-flow, Presentation, Security	Validate `response_uri`	Before sending the Authorization Response, Wallet Instance confirms the `response_uri` exactly matches one of the Relying Party's attested `response_uris` in its metadata; a mismatch aborts the request.
WP_092	Remote-flow, Presentation, Security	Encrypt Authorization Response	Wallet Instance encrypts the Authorization Response JWT per Section 8.3 of [OpenID4VP] using the Relying Party's public key.
WP_093	Remote-flow, Presentation, Security	Construct `vp_token` with `state`	The Authorization Response JWT payload includes the original `state` value and a `vp_token` object with entries for each requested Credential presentation.
WP_093a	Remote-flow, Presentation, Interoperability	Include signed presentations	Within that `vp_token`, there are the requested Credential(s) in SD JWT VC format.
WP_093b	Remote-flow, Presentation, Security	Append Key Binding JWT	Wallet Instance generates for every individual SD-JWT presentation within the `vp_token`, its own corresponding Key Binding JWT.
WP_093c	Remote-flow, Presentation, Security	Key Binding JWT format	The JOSE header of each generated Key Binding JWT includes `"typ":"kb+jwt"` and an `alg` to one of the supported signature algorithms, as well as a payload including the `iat`, `aud`, `nonce`, and `sd_hash` claims per OpenID4VP .
WP_094	Remote-flow, Presentation, Interoperability	Perform user-agent redirect	Upon receiving the `redirect_uri`, Wallet Instance successfully performs a user-agent redirect to the `redirect_uri` supplied by the Relying Party, so the Relying Party can resume the interaction on the same device that initiated the flow.
WP_094a	Remote-flow, Trust, Presentation, Security	Validate `redirect_uri`	Wallet Instance verifies that the `redirect_uri` exactly matches one of the URIs listed in the Relying Party's `redirect_uris` metadata. If it does not match, aborts the flow.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_095	Proximity-flow, Presentation, Security	Support supervised/ unsupervised retrieval	Wallet Instance supports a Credential presentation using the Supervised Device Retrieval method for an in-person verification scenario and via Device Retrieval (unsupervised) for automated verification without human oversight.
WP_096	Proximity-flow, Presentation, Security	Device retrieval support	Wallet Instance supports Device Retrieval mechanisms via the Bluetooth Low Energy (BLE) or NFC.
WP_096a	Proximity-flow, Presentation, Security	Enforce device retrieval only	Wallet Instance rejects any request to initiate a proximity flow using the Server Retrieval mechanism.
WP_096b	Proximity-flow, Presentation, Security	Support BLE/ NFC retrieval (conditional)	Wallet Instance completes the BLE or NFC retrieval flow successfully. BLE support is mandatory, while NFC support is RECOMMENDED. Support for at least one device retrieval method (BLE or NFC) is mandatory; failure constitutes non-compliance.
WP_097	Proximity-flow, Presentation, UX	Supported DeviceEngagement mechanisms	Wallet Instance supports DeviceEngagement based on QR code or NFC Connection Handover.
WP_097a	Proximity-flow, Presentation, Security	Support QR/ NFC engagement (conditional)	Wallet Instance completes a full data-retrieval transaction over BLE and over NFC (where hardware is available). QR code support is mandatory, while NFC support is RECOMMENDED. Support for at least one engagement method (QR or NFC) is mandatory; failure constitutes non-compliance.
WP_098	Proximity-flow, Presentation, Security	Relying Party authentication	Wallet Instance supports and performs Relying Party Instance Authentication in accordance with ISO18013-5 reader-authentication.
WP_099	Proximity-flow, Presentation, Interoperability	Domestic mDL support	Wallet Instance supports and processes mDL Credentials using both the IT Wallet domestic document type and namespaces and the standard ISO18013-5 definitions without error (see mdoc-CBOR Credential Format for more details).

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_100	Proximity-flow, Presentation, Security	WSCA authentication	Wallet Instance prompts the User to perform a WSCA-based authentication, directly or by unlocking the application, and does not proceed with the proximity flow until it succeeds.
WP_101	Proximity-flow, Presentation, Security	Generate ephemeral EC key pair	Wallet Instance successfully generates a fresh ephemeral elliptic-curve key pair (per the chosen ISO18013-5 cipher suite).
WP_102	Proximity-flow, Presentation, Interoperability	CBOR encoding of Device Engagement data	DeviceEngagement is CBOR encoded and contains Version, Security, Capabilities, and OriginInfos (may be empty). For QR engagement it also includes DeviceRetrieval Mode-BLEOptions and/or DeviceRetrieval Mode-NFCOptions. For NFC engagement, these options are omitted.
WP_102a	Proximity-flow, Presentation, Interoperability	DeviceEngagement over QR	The QR mdoc: URI encodes the valid CBOR DeviceEngagement encoded (per Section 9. 1 of [ISO18013-5]) using base64url-without-padding (**RFC 4648**).
WP_102b	Proximity-flow, Presentation, Interoperability	Verify Security component	The Security component within the Device Engagement data contains a supported cipher suite identifier and the Wallet Instance's ephemeral public key per ISO18013-5 Table 22.
WP_102c	Proximity-flow, Presentation, Interoperability	Verify DeviceRetrievalMode-BLEOptions component	The DeviceRetrievalMode-BLEOptions component within the DeviceEngagement data indicates Central Client Mode and carries the correct device UUID.
WP_102d	Proximity-flow, Presentation, Interoperability	Verify DeviceRetrievalMode-NFCOptions component	The DeviceRetrievalMode-NFCOptions component declares supported role (PICC for Wallet Instance) and maximum APDU command/response sizes per ISO mapping.
WP_102e	Proximity-flow, Presentation, Interoperability	Verify Capabilities component	The Capabilities component within the Device Engagement data correctly sets both the `HandoverSessionEstablishmentSupport` and `ReaderAuthAllSupport` flags to `true`.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_102f	Proximity-flow, Presentation, Interoperability	Verify OriginInfos component	If the OriginInfos component, is present within the DeviceEngagement data, is correctly encoded as an array (which may be empty).
WP_103	Proximity-flow, Presentation, UX	DeviceEngagement over NFC (Connection Handover)	Wallet Instance exposes DeviceEngagement via NFC Connection Handover either Static or Negotiated.
WP_103a	Proximity-flow, Presentation, Interoperability	NFC Connection Handover (Static)	Wallet Instance acts as NFC Tag (Type 4) and provides a Handover Select with at least one Alternative Carrier Record and associated Carrier Configuration Record; an Auxiliary Data Record carries the DeviceEngagement (type iso.org:18013:deviceengagement, id "mdoc").
WP_103b	Proximity-flow, Presentation, Interoperability	NFC Connection Handover (Negotiated)	Wallet Instance exposes the service `urn:nfc:sn:handover`; upon Handover Request, it returns Handover Select with exactly one selected carrier and the auxiliary DeviceEngagement.
WP_103c	Proximity-flow, Presentation, Security, Interoperability	Early SessionEstablishment via TNEP	If `HandoverSessionEstablishmentSupport` set to `true` in DeviceEngagement, Wallet Instance accepts early `SessionEstablishment` over the announced TNEP service during negotiated handover and later verifies it matches the `SessionEstablishment` received during data retrieval.
WP_103d	Proximity-flow, Presentation, Interoperability	Carrier Configuration Record encoding	Wallet Instance successfully includes a Carrier Configuration Record for each supported carrier. For NFC, type `iso.org:18013:nfc`, ID `nfc`, content per Section 9.2.2 of [ISO18013-5]. For BLE, content per Section 11.1.2 of [ISO18013-5].
WP_103e	Proximity-flow, Presentation, Interoperability	Auxiliary Data Record for DeviceEngagement	Wallet Instance successfully provides an Auxiliary Data Record carrying DeviceEngagement with type `iso.org:18013:deviceengagement`, ID `mdoc`, using NFC Forum external type (0x04). Each Alternative Carrier Record MUST reference this record.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_103f	Proximity-flow, Presentation, Interoperability	Alternative Carrier Records	In Static Handover, Handover Select includes one oe more Alternative Carrier Records; in Negotiated Handover, Handover Select contains exactly one selected carrier. The NFC Alternative Carrier references the `nfc` Carrier Configuration Record.
WP_103g	Proximity-flow, Presentation, Security	Early SessionEstablishment mismatch	If early SessionEstablishment is received during Negotiated Handover, Wallet Instance successfully verifies that it matches the SessionEstablishment received during Device Retrieval; on mismatch Wallet Instance terminates the flow
WP_103h	Proximity-flow, Presentation, Interoperability	No early SessionEstablishment	If no early SessionEstablishment is received during Negotiated Handover, the Wallet Instance proceeds with Device Retrieval as normal (no failure), per the NFC DeviceEngagement note.
WP_104	Proximity-flow, Presentation, Security	Derive session keys	Wallet Instance successfully derives session keys by performing the negotiated key-agreement protocol with its ephemeral private key and the Relying Party's ephemeral public key.
WP_105	Proximity-flow, Presentation, Security	Decrypt & verify `SessionEstablishment`	Wallet Instance successfully decrypts the `SessionEstablishment` message using the derived session key and validates the Relying Party's signature.
WP_106	Proximity-flow, Presentation, Security	Validate `SessionEstablishment` contents	Wallet Instance verifies the `SessionEstablishment` message includes the Relying Party's Pub Key and a request for specific attribute(s) from the Relying Party.
WP_107	Proximity-flow, Presentation, Privacy	Prompt attribute consent	Wallet Instance decrypts and displays the requested attributes to the User in a consent screen and proceeds only after receiving explicit User approval.
WP_107a	Proximity-flow, Presentation, Privacy	Display Relying Party certificate	Wallet Instance displays the full, parsed Relying Party Registration Certificate to the User for transparency before User consent and data disclosure.
WP_108	Proximity-flow, Presentation, Interoperability	Retrieve mdoc Credentials	Wallet Instance successfully retrieves each requested mdoc Credential from storage, preparing it for the mdoc Response.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_109	Proximity-flow, Presentation, Interoperability	Prepare mdoc Response	Wallet Instance successfully builds the CBOR-encoded `SessionData` message (the mdoc Response) including a `documents` array populated with the requested Credentials.
WP_110	Proximity-flow, Presentation, Security, Interoperability	mdoc authentication	Wallet Instance correctly signs the `deviceSigned` authentication data for each presented Credential, following the mdoc authentication process as specified in Section 12.4 of [ISO18013-5].
WP_111	Proximity-flow, Presentation, Interoperability	Validate mdoc Response structure	Wallet Instance constructs the mdoc Response as a CBOR-encoded object that includes a `version` component at its root and ensures each document within the `documents` array contains the mandatory `docType` component.
WP_111a	Proximity-flow, Presentation, Security	Validate `deviceSigned` component	Within each document, the `deviceSigned` component includes a `deviceNameSpaces` structure (possibly empty) plus a `deviceAuth` `COSE_Sign1` containing the required `deviceSignature` over the device-authentication data.
WP_112	Proximity-flow, Presentation, Security	Encrypt `SessionData`	Wallet Instance encrypts the `SessionData` message with the derived session keys.
WP_112a	Proximity-flow, Presentation, Interoperability	BLE transport of messages	Over BLE (using the ISO GATT characteristics), Wallet Instance correctly receives the Relying Party's encrypted `SessionEstablishment` and transmits its encrypted `SessionData` (and any status/termination codes).
WP_112b	Proximity-flow, Presentation, Interoperability	NFC transport of messages	Over NFC (using the APDU flow), the Wallet Instance correctly receives the Relying Party's encrypted `SessionEstablishment` and transmits its encrypted `SessionData`, handling status words and fragmentation without errors.
WP_112c	Proximity-flow, Presentation, Interoperability	BLE UUID/MTU consistency	The Wallet's BLE service/characteristic *UUIDs* and effective MTU used on the link are consistent with the values advertised in Device Engagement (and/or Carrier Configuration), and the Wallet segments messages accordingly.

continues on next page

Table 20.29 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
WP_112d	Proximity-flow, Presentation, Security	BLE Ident characteristic (optional)	If the Ident characteristic is implemented, Wallet Instance verifies the Relying Party's Ident value before data retrieval and terminates the BLE connection on mismatch, per Section 11.1.3 of [ISO18013-5].
WP_112e	Proximity-flow, Presentation, Interoperability	NFC SELECT AID handling	Wallet Instance successfully exposes *AID* and, upon SELECT *APDU*, returns valid *FCI* with correct *SW1/SW2*; subsequent ENVELOPE/ GET RESPONSE exchanges.
WP_112f	Proximity-flow, Presentation, Interoperability	NFC APDU size consistency	The Wallet's *APDU* command/response sizing (including fragmentation and SW1=61 handling) is consistent with the max *PDU* sizes it advertised (Carrier Configuration / NFCOptions) during Device Engagement.
WP_113	Proximity-flow, Presentation, Security	Terminate session/ inactivity timeout	Wallet Instance automatically terminates the session if no session-related messages are sent or received for the duration of its configured inactivity timeout.
WP_113a	Proximity-flow, Presentation, Security	Terminate session/ no further request	Wallet Instance terminates the session once either it or the Relying Party has concluded the data exchange.
WP_113b	Proximity-flow, Presentation, Security	Send termination signal	Wallet Instance transmits End command (or termination code) over BLE to signal session closure.
WP_113c	Proximity-flow, Presentation, Security	Send termination over NFC	Wallet Instance signals end via a status code in `SessionData` and/or completes the APDU exchange per ISO; no further APDUs are accepted for the session.
WP_114	Proximity-flow, Presentation, Security	Destroy session keys	When a session is terminated, Wallet Instance wipes all session keys and related ephemeral material from memory and storage.
WP_114a	Proximity-flow, Presentation, Security	Close communication channel	Wallet Instance closes the active communication channel by disconnecting the BLE link or ending the NFC APDU exchange.

Test Cases for User Attribute Deletion on Relying Party Side

This section lists the test cases from Sections:

- *User's Attributes Deletion* (page 297)
- Relying Party Provider Backend Erasure Endpoint

Test Case ID	Purpose	Description	Expected Results
WP_115	Attribute Deletion, Lifecycle, Privacy	Attribute deletion function	Wallet Instance UI provides functions for the User to request attribute deletion, view transaction logs, and see a list of Relying Parties that hold their attributes.
WP_115a	Attribute Deletion, Lifecycle, Privacy	Transaction logs for attribute deletion	The transaction log view shows Relying Parties in possession of the User's attributes.
WP_116	Attribute Deletion, Lifecycle, Security	Relying Party metadata validation for deletion	When a User selects a Relying Party for attribute deletion, Wallet Instance fetches and validates its federation metadata, confirming the presence of an Erasure Endpoint.
WP_117	Attribute Deletion, Lifecycle, Interoperability	Erasure Request	Wallet Instance correctly constructs, and sends a valid Erasure Request destined for the Relying Party's Erasure Endpoint.
WP_117a	Attribute Deletion, Lifecycle, Privacy	Erasure Request logging	A log entry is created for each Erasure Request, containing the timestamp, the Relying Party's identifier, and the specific attributes requested for deletion.
WP_118	Attribute Deletion, Lifecycle, UX	User redirection and callback for erasure	Wallet Instance successfully redirects the User to the Relying Party's Erasure Endpoint and receives the Erasure Response via its callback mechanism.
WP_119	Attribute Deletion, Lifecycle, Interoperability	Handles deletion/error response	Wallet Instance correctly processes both success and error responses for the Erasure Response from the Relying Party.
WP_119a	Attribute Deletion, Lifecycle, UX	User notification on erasure	Wallet Instance displays a clear notification to the User indicating the success or failure of the Erasure Request.

Test Cases for Digital Credential's Backup and Restore

This section lists the test cases from Section *Backup and Restore* (page 300).

Test Case ID	Purpose	Description	Expected Results
WP_120	Backup and Restore, UX	Credential backup initiation	The User successfully triggers the Credential backup operation on Wallet Instance.
WP_120a	Backup and Restore, Security	Random key phrase display	Wallet Instance invokes the backup APIs, selects 10 random key phrases from a pre-generated list and displays them to the User.
WP_120b	Backup and Restore, Security	Secure key phrase storage instruction	Wallet Instance displays a clear instruction for the User to securely store the generated key phrases.
WP_121	Backup and Restore, Security	Key derivation from key phrases	Wallet Instance derives an encryption key from the User's key phrases using a key derivation function (e.g., `PBKDF2`, `Bcrypt`, `Scrypt`, `Argon2`).
WP_121a	Backup and Restore, Security	Key derivation function configuration	If `PBKDF2` with `SHA-256` is used, the function is configured with an iteration count of at least `600,000`.
WP_122	Backup and Restore, Interoperability	Signed backup JWT creation	Wallet Instance creates a backup file containing a signed JWT that encapsulates the backup data.
WP_122a	Backup and Restore, Interoperability	JOSE header for backup JWT	The backup JWT header contains the correct `alg` and a `typ` of `wallet-unit-credentials-backup+jwt`.
WP_122b	Backup and Restore, Interoperability	Backup JWT payload contents	The backup JWT payload includes the required claims: `timestamp`, `wallet_provider_id`, `wallet_instance_version`, `wallet_attestation`, and `credentials_backup`.

continues on next page

Table 20.31 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_122c	Backup and Restore, Security	Optional transaction history in backup JWT	The `credentials_backup` claim contains the transaction history for each Credential, if supported.
WP_122d	Backup and Restore, Security	Credential identifiers in backup JWT	The `credentials_backup` claim contains entries keyed by the Issuer identifier, and each entry lists one or more `credential_configuration_ids` corresponding to the issued credentials.
WP_123	Backup and Restore, Security	Backup JWT signing with attested key	Wallet Instance signs the backup JWT with the private key corresponding to the public key found in the `cnf` claim of the Wallet Instance Attestation (WIA) JWT.
WP_123a	Backup and Restore, Security	Attestation validity check for backup JWT	Before signing the backup JWT, Wallet Instance verifies the validity of its own Wallet Instance Attestation (WIA).
WP_124	Backup and Restore, Security	Backup file encryption with User key	Wallet Instance encrypts the backup using the key derived from the User's key phrases.
WP_125	Backup and Restore, UX	Secure storage location selection prompt	Wallet Instance prompts the User to select a location to save the encrypted backup file, offering options such as local or cloud storage.
WP_126	Backup and Restore, UX	Credential restore initiation	The User successfully triggers the Credential restore operation on Wallet Instance.
WP_127	Backup and Restore, UX	Backup file/key phrase selection for restore	The User successfully selects a backup file from the storage and enters their recovery key phrases.
WP_128	Backup and Restore, Interoperability	Wallet Instance Attestation (WIA) extraction from backup JWT	Wallet Instance extracts the Wallet Instance Attestation (WIA) JWT from the `wallet_attestation` claim of backup JWT found within the backup file.
WP_128a	Backup and Restore, Security	Ignore Wallet Instance Attestation (WIA) expiration during restore	The restore process continues successfully even if the Wallet Instance Attestation (WIA) JWT in the backup file is expired.

continues on next page

Table 20.31 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_129	Backup and Restore, Security	Backup JWT signature verification	Wallet Instance successfully verifies the signature of the backup JWT using the Wallet Instance Attestation (WIA) public key from the `cnf.jwk` claim within the `wallet_attestation` claim.
WP_130	Backup and Restore, Interoperability	Credential issuance from restored backup	For each Credential entry in the backup JWT, Wallet Instance successfully extracts the Issuer URL, and the `credential_configuration_id`.
WP_130a	Backup and Restore, Interoperability	Retrieve the Issuer metadata	For each Issuer URL, Wallet Instance successfully fetches the corresponding Issuer metadata.
WP_130b	Backup and Restore, Interoperability	Credential issuance request	For each Credential, Wallet Instance successfully initiates a new Wallet-Initiated Authorization Code Issuance Flow bound to the new Wallet Instance key (fresh Holder Key Binding), and not a token renewal or a Re-Issuance Flow request.

Optional Test Cases for Wallet Instance

This section lists the test cases from Sections:

- *Wallet Instance* (page 279)
- Request-Specific Error Responses
- Mobile Application Instance Initialization

These test cases are optional and have been designed for the IT Wallet implementation, managed by PagoPA. They are intended for implementers who need to verify compatibility with the specific features of this reference model.

Test Case ID	Purpose	Description	Expected Results
WP_131	Wallet Initialization / Registration, Lifecycle, Security	Nonce request for replay protection	Wallet Instance requests and receives a single-use, short-lived nonce from the Wallet Provider's Nonce Endpoint.
WP_132	Wallet Initialization / Registration, Lifecycle, Interoperability	Hardware-backed EC key pair generation/storage	Wallet Instance successfully generates a fresh hardware-backed EC key pair and stores its associated `hardware_key_tag` in the local secure storage.
WP_132a	Wallet Initialization / Registration, Lifecycle, Security	Pre-existing key deletion during initialization	Wallet Instance checks for the presence of pre-existing Cryptographic Hardware Keys during initialization, and successfully deletes them.
WP_133	Wallet Initialization / Registration, Lifecycle, Interoperability	Wallet Initialization Request submission	Wallet Instance successfully sends to the Wallet Provider an Initialization request with the required `nonce`, `key_attestation`, and `hardware_key_tag` parameters.
WP_133a	Wallet Initialization / Registration, Lifecycle, Interoperability	SHA-256 hash computation	Wallet Instance successfully computes a SHA-256 digest (`client_data_hash`) over the `nonce`, the `hardware_key_pub`, and the `hardware_key_tag`,
WP_133b	Wallet Initialization / Registration, Lifecycle, Interoperability	Key Attestation	Wallet Instance successfully invokes the Key Attestation API with the `client_data_hash` and obtains a signed attestation from the Key Attestation API.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_134	Wallet Initialization / Registration, Lifecycle, Security	Key Binding request	Wallet Instance sends a Key Binding request that contains an `assertion` parameter whose value is a signed JWT.
WP_134a	Wallet Initialization / Registration, Lifecycle, Security	Signing Key Binding JWT	Wallet Instance successfully signs the Key Binding request JWT with the Wallet Hardware public key.
WP_135	Wallet Initialization / Registration, Lifecycle, Security	Wallet Initialization Request validation	Wallet Provider successfully checks the Initialization Request parameters: the `nonce` and `key_attestation`, and, if valid, returns a confirmation response.
WP_135a	Wallet Initialization / Registration, Lifecycle, Security	Nonce uniqueness verification	Wallet Provider rejects any Initialization Request containing a nonce that it did not generate or that has already been used.
WP_135b	Wallet Initialization / Registration, Lifecycle, Security	Key Attestation validation per guidelines	Wallet Provider validates the `key_attestation` against manufacturer guidelines and confirms the device meets its minimum security requirements.
WP_136	Wallet Initialization / Registration, Lifecycle, Security	Cryptographic binding verification	Wallet Provider successfully verifies the cryptographic binding between the `hardware_key_tag`, `hardware_key_pub`, nonce, and the `client_data_hash` provided in the key_attestation.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_137	Wallet Initialization / Registration, Lifecycle, Interoperability	Wallet Instance Registration	Wallet Provider successfully registers Wallet Instance, storing its `hardware_key_tag`, `hardware_key_pub`, and other device information.
WP_138	Wallet Initialization / Activation, Lifecycle, Security	User account creation	Wallet Provider creates a User account associated with Wallet Instance's `hardware_key_tag`.
WP_139	Wallet Uninstallation, Lifecycle, Security	Key/state deletion on uninstallation	Wallet Instance uninstalls itself and removes all local keys and application data.
WP_140	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Wallet Instance Attestation (WIA) Request	Wallet Instance successfully constructs the Wallet Instance Attestation (WIA) Request JWT with the required claims: `integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, and other configuration related parameters.
WP_140a	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Hardware key existence check/ re-initialization	Wallet Instance checks for the existence of Cryptographic Hardware Keys; if they are not found, it triggers the re-initialization process.
WP_140b	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Interoperability	Requests nonce for Wallet Instance Attestation (WIA) Request	Wallet Instance successfully requests and receives a fresh nonce from the Wallet Provider's Nonce Endpoint, before requesting the Wallet Instance Attestation (WIA).

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_140c	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Interoperability	Computes hash for Wallet Instance Attestation (WIA) Request	Wallet Instance computes a SHA-256 digest (`client_data_hash`) of the nonce and the thumbprint of the `ephemeral_key_pub` JWK.
WP_140d	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Interoperability	Signs hash with hardware private key	Wallet Instance signs the `client_data_hash` with the hardware private key and produces a valid `hardware_signature`.
WP_140e	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Interoperability	Obtains signed Integrity Assertion from Device Integrity Service.	Wallet Instance successfully requests and receives a signed `integrity_assertion` from the Device Integrity Service that is bound to the `client_data_hash`.
WP_140f	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Include `cnf` claim	The `cnf` claim contains the ephemeral public key, linking the key to the attestation.
WP_141	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Signing JWT Wallet Instance Attestation (WIA) Request	Wallet Instance signs the Wallet Instance Attestation (WIA) Request JWT with the ephemeral private key.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_142	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Submitting Wallet Instance Attestation (WIA) Request JWT to Wallet Provider	The Wallet Instance submits the signed request JWT as an `assertion` parameter to the Wallet Provider's Attestation Issuance Endpoint.
WP_143	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Attestation Request JWT verification	Wallet Provider successfully performs a comprehensive validation of the Wallet Instance Attestation (WIA) Request JWT, including its signature, claims, and cryptographic proofs.
WP_143a	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	HTTP header validation for Wallet Instance Attestation (WIA) Request	Wallet Provider successfully validates the Wallet Instance Attestation (WIA) Request JWT header to contain valid `alg`, `kid`, and `typ` parameters.
WP_143b	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	JWT signature verification in Wallet Instance Attestation (WIA) Request	Wallet Provider successfully verifies the signature of the Wallet Instance Attestation (WIA) Request JWT using the public key in the provided JWK.
WP_143c	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Nonce uniqueness verification for Wallet Instance Attestation (WIA)	Wallet Provider rejects the Wallet Instance Attestation (WIA) Request JWT if the nonce was not generated by itself or has been previously used.
WP_143d	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Registered Wallet Instance verification	Wallet Provider confirms that the Wallet Instance Attestation (WIA) Request originates from a valid and currently registered Wallet Instance; if not rejects the request.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_143e	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Interoperability	Hardware signature validation	Wallet Provider successfully reconstructs the client_data, and validates the `hardware_signature` using Wallet Instance's registered Hardware public key.
WP_143f	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Integrity Assertion validation per guidelines	Wallet Provider successfully validates the `integrity_assertion` according to the device manufacturer's guidelines.
WP_143g	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	`iss` parameter verification	Wallet Provider verifies that the `iss` parameter in the Wallet Instance Attestation (WIA) Request JWT matches its own URL identifier.
WP_144	Wallet Instance Attestation (WIA) Issuance, Lifecycle, Security	Attestation Issuance	After successful validation of the Wallet Instance Attestation (WIA) Request, the Wallet Provider issues a Wallet Instance Attestation (WIA) that meets *the Wallet Instance Attestation lifetime* (page 272).
WP_145	Wallet Revocation, Lifecycle, Interoperability	Wallet Instance status retrieval	Wallet Instance sends a Wallet Instances Retrieval Request to the Wallet Provider's Wallet Instance Management Endpoint.
WP_146	Wallet Revocation, Lifecycle, Interoperability	Wallet Instance status response	Wallet Provider validates the request and returns Wallet Instance a list of the authenticated User's linked Wallet Instances, including each instance's `ID`, `status`, and `issued_at` fields.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_147	Wallet Revocation, Lifecycle, Interoperability	Wallet Instance Revocation Request body	Wallet Instance Revocation Request includes a body with `{"status":"REVOKED"}`.
WP_148	Wallet Revocation, Lifecycle, Interoperability	Wallet Instance Revocation Response	Wallet Provider updates the specified Wallet Instance state to `REVOKED` and returns a confirmation.
WP_149	Wallet Revocation, Lifecycle, Security	Clears keys upon revocation	As part of its revocation, Wallet Instance removes all associated Wallet Cryptographic Hardware Key pairs from the device.
WP_150	Wallet Initialization / Registration, Lifecycle, Interoperability	Initialization failure (invalid nonce)	When a Wallet Instance Initialization request contains a nonce that is invalid, expired, or already used, Wallet Provider returns a `403 Forbidden` response with the error code `invalid_request`.
WP_151	Wallet Initialization / Registration, Lifecycle, Interoperability	Initialization failure (Key Attestation signature)	When a Wallet Instance Initialization request contains an invalid Key Attestation signature, Wallet Provider returns a `403 Forbidden` response with the error code `invalid_request`.
WP_152	Wallet Initialization / Registration, Lifecycle, Interoperability	Key Binding failure (Wallet Instance not found)	When a Key Binding request references a Wallet Instance that is not found, Wallet Provider returns a `404 Not Found` response with the error code `not_found`.
WP_153	Wallet Initialization / Registration, Lifecycle, Interoperability	Key Binding failure (Wallet Instance revoked)	When a Key Binding request is made for a Wallet Instance that has been revoked, Wallet Provider returns a `403 Forbidden` response with the error code `invalid_request`.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_154	Wallet Initialization / Registration, Lifecycle, Interoperability	Key Binding failure (invalid PoP)	When a Key Binding request contains an invalid Proof of Possession (`hardware_signature`), Wallet Provider returns a `403 Forbidden` response with the error code `invalid_request`.
WP_155	Wallet Initialization / Registration, Lifecycle, Interoperability	Key Binding failure (Integrity Assertion failure)	When the Integrity Assertion in a Key Binding request fails validation (e.g., is tampered with), Wallet Provider returns a `403 Forbidden` response with the error code `invalid_request`.
WP_156	Trust, Security	WUA distinct from WIA	Wallet Unit Attestation (WUA) is issued and verified as a distinct artefact from the WIA. Private keys bound to a WUA are stored in the same Keystore or WSCD as the Digital Credential keys that the WUA attests. Tests MUST NOT use the legacy name «Wallet Attestation» for either artefact.
WP_157	Issuance, Authentication	IT-Wallet ID issuance authentication	The Wallet Instance obtains an IT-Wallet ID using CieID LoA High as the primary method, or the eID Substantial Authentication with MRTD Verification flow as the alternative. The flow is distinct from PID High issuance.
WP_158	Issuance, Authentication	L2+ `authorization_details`	When requesting an IT-Wallet ID with eID Substantial Authentication with MRTD Verification, the PAR Request Object includes an `authorization_details` object with `type` equal to `it_l2+document_proof` and the claims required by *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457). If that object is absent, the EAA Provider authenticates the User with CieID LoA High.

continues on next page

Table 20.32 – continued from previous page

Test Case ID	Purpose	Description	Expected Results
WP_159	Issuance, Authentication, Security	L2+ IdP–MRTD identity binding	After primary authentication (SPID or CieID Substantial), the EAA Provider correlates the identity in the IdP assertion with the identity in the MRTD (CIE). The comparison uses at least the tax identification number or an equivalent unique identifier present in both sources. A mismatch aborts issuance. If primary authentication is LoA High, Phase 3 (MRTD PoP) is skipped.
WP_160	Presentation, Security	Offline presentation and key location	Offline or proximity presentation of a Digital Credential succeeds only when the bound private keys are in a local Keystore or local WSCD. Credentials whose keys are in a remote WSCD are not presented offline.
WP_161	Trust, Security	Authenticated PDND Wallet Provider endpoints	Calls to the e-Service PDND Wallet Provider Catalogue, including Notify User Death, are authenticated and authorised. Unauthenticated requests are rejected. A successful Notify User Death leads to Wallet Instance revocation as for other PID/IT-Wallet ID Provider-initiated revocations.

Credential Issuer Test Matrix

This section provides the set of test cases designed for technical implementers and development teams responsible for creating and deploying Credential Issuer solutions. It is also intended for assessment bodies inspecting and validating the implementations of Credential Issuer solutions.

Test Case ID	Purpose	Description	Expected Result
CI_001	Trust, Security	Entity Configuration publication	Federation Entity publishes its own Entity Configuration in the .well-known/openid-federation endpoint.
CI_002	Trust, Interoperability	Entity Configuration response media type check	The Entity Configuration HTTP Response is set to media type to application/entity-statement+jwt.
CI_003	Trust, Security	Entity Configuration cryptography	The Entity Configuration is cryptographically signed

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_004	Trust, Security	Public Key inclusion in Entity Configuration and Subordinate Statement	The Entity Configuration and the Subordinate Statement issued by the immediate superior both include the public part of the key.
CI_005	Trust, Security	Entity Configuration's Trust Marks	The Entity Configuration contains one or more Trust Marks.
CI_006	Trust, Interoperability	Entity Configurations parameters	Entity Configurations have in common these parameters: iss, sub, iat, exp, jwks, metadata.
CI_007	Voided	Voided	Voided
CI_008	Trust, Interoperability	Credential Issuer metadata	Credential Issuer successfully provides the following metadata types: *federation_entity*, *Oauth_authorization_server* and *openid_credential_issuer*
CI_009	Trust, Interoperability	Inclusion of *openid_credential_verifier* Metadata in User Authentication via Wallet	When the (Q)EAA Providers authenticate users through their Wallet Instance, the *openid_credential_verifier* metadata is included in addition to the required metadata parameters.
CI_010	Issuance, Interoperability	Credential Offer URI Structure	Credential Issuer generates a Credential Offer consisting of a single URI query parameter named "credential_offer"ensuring the URL is well-formed and contains only this parameter for the offer data.
CI_011	Issuance, Interoperability	Credential Offer Delivery Methods	Credential Offer URL is successfully embedded in QR codes or included as clickable href buttons in HTML pages.
CI_012	Issuance, Interoperability	Credential Offer Mandatory Parameters	Credential Offer contains all three mandatory parameters (credential_issuer, credential_configuration_ids, and grants) with valid values

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_013	Issuance, Interoperability	Credential Offer Grants Parameter Structure	The grants parameter successfully contains an `authorization_code` object that includes both required sub-parameters (`issuer_state` and `authorization_server`) with appropriate values.
CI_014	Issuance, Interoperability	Credential Object Compilation	Credential Issuer ensures the Credential Object is properly compiled with all required fields, correct formatting, and valid data structures before delivery
CI_015	Issuance, Security	PAR Request Object Signature Validation	Credential Issuer successfully validates the Request Object signature
CI_015a	Issuance, Security	PAR Request Object Algorithm Header Processing	Credential Issuer uses the algorithm specified in the alg header parameter (**RFC 9126**/**RFC 9101**) to validate the Request Object Signature
CI_015b	Issuance, Security	PAR Wallet Instance Attestation (WIA) Public Key Retrieval	Credential Issuer successfully retrieves the public key from the Wallet Instance Attestation (WIA)'s cnf.jwk claim
CI_015c	Issuance, Security	PAR JWT Key Identifier Reference	The Credential Issuer successfully references the correct key via the kid JWT header parameter
CI_015d	Issuance, Security	PAR Cryptographic Signature Integrity Validation	The Credential Issuer successfully complete cryptographic signature integrity before proceeding with any further validations
CI_016	Issuance, Interoperability	PAR HTTP POST Parameter Encoding	Credential Issuer successfully processes HTTP POST requests with message body parameters encoded in application/x-www-form-urlencoded format;

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_017	Issuance, Interoperability	PAR Scope and Authorization Details Interpretation	The Pushed Authorization Request contains both the scope value and the authorization_details parameter. The Credential Issuer processes each parameter independently.
CI_017a	Issuance, Interoperability	PAR Authorization Details	When both scope and authorization_details request the same Credential type, Credential Issuer follows the specifications given by the authorization_details object
CI_018	Voided	Voided	Voided
CI_019	Issuance, Security	PAR Algorithm Compliance Check	Credential Issuer verifies the signing algorithm in the alg header matches one of the approved algorithms listed in the Cryptographic Algorithms section; rejects requests with non-compliant algorithms and returns appropriate error.
CI_020	Issuance, Security	PAR Client ID Consistency Validation	Credential Issuer confirms the client_id in the PAR request body exactly matches the client_id claim in the Request Object; mismatched values trigger request rejection with clear error message.
CI_021	Issuance, Security	Issuer-Client ID Matching in the PAR request	Credential Issuer validates that the iss claim in the Request Object matches the client_id claim within the same Request Object (**RFC 9126**/**RFC 9101**); discrepancies result in request denial.
CI_022	Issuance, Security	PAR Audience Claim Verification	Credential Issuer confirms the aud claim in the Request Object equals the Credential Issuer's own identifier (**RFC 9126**/**RFC 9101**); incorrect audience values cause immediate request rejection.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_023	Issuance, Security	PAR Request URI Parameter Rejection	Credential Issuer detects and rejects any PAR request containing the request_uri parameter (**RFC 9126**), returning an appropriate error response indicating the parameter is not supported.
CI_024	Issuance, Security	PAR Mandatory Parameters Validation	Credential Issuer verifies all mandatory HTTP parameters are present in the Request Object and validates their values against the defined Table specifications (derived from **RFC 9126**); missing or invalid parameters trigger structured error responses.
CI_025	Issuance, Security	PAR Token Expiration Check	Credential Issuer validates the Request Object has not expired by checking the exp claim against current time; expired tokens are rejected with timestamp-related error messages.
CI_026	Issuance, Security	PAR Token Issuance Time Validation	Credential Issuer confirms the iat claim represents a past timestamp
CI_026a	Issuance, Security	Recommended PAR Token Issuance Time rejection	Credential Issuer reject requests where iat is more than 5 minutes from current time (**RFC 9126**); timing violations result in "invalid_request" errors.
CI_027	Issuance, Security	PAR Replay Attack Prevention	Credential Issuer successfully checks that the jti claim in the Request Object has not been used before by the Wallet Instance identified by the client_id.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_028	Issuance, Security, Interoperability	OAuth Client Attestation PoP Validation	.Credential Issuer successfully validates the OAuth-Client-Attestation-PoP parameter according to Section 5 of [OAUTH-ATTESTATION-CLIENT-AUTH], confirming proof-of-possession and rejecting invalid attestations with detailed error responses.
CI_029	Issuance, Trust	Wallet Instance Trustworthiness Verification	Credential Issuer successfully verifies the trustworthiness and indirect Federation membership of the Wallet Instance through comprehensive Wallet Instance Attestation (WIA) validation
CI_030	Issuance, Trust	Wallet Provider Federation Membership Validation	Credential Issuer confirms the Wallet Provider (issuer of the Wallet Instance Attestation (WIA)) is a recognized and trusted Federation member by checking Federation registries and trust lists;
CI_031	Issuance, Security	Wallet Instance Attestation (WIA) Cryptographic Signature Validation	Credential Issuer successfully validates the cryptographic signature of the Wallet Instance Attestation (WIA) using the Wallet Provider's public key, ensuring signature integrity and authenticity;
CI_032	Issuance, Security	Wallet Instance Attestation (WIA) Expiration Check	Credential Issuer verifies the Wallet Instance Attestation (WIA) has not expired at verification time by checking timestamp claims against current time
CI_033	Issuance, Security	Wallet Instance Attestation (WIA) Attested Cryptographic Key Acceptance	Credential Issuer accepts and uses only cryptographic keys that are properly derived from the attested Wallet Instance during the credential issuance process;

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_034	Issuance, Security	Wallet Instance Attestation (WIA) Device Security and Compliance Verification	Credential Issuer relies on Wallet Instance Attestation (WIA) claims to confirm the Wallet Instance operates on a secure, trusted device that meets the Issuer's required security standards;
CI_035	Issuance, Trust	Wallet Provider Trust Chain Evaluation	Credential Issuer successfully evaluates the complete Trust Chain of the Wallet Instance Attestation (WIA)'s issuer (Wallet Provider)
CI_036	Issuance, Trust, Interoperability	Federation Metadata Retrieval	Credential Issuer successfully uses Federation API endpoints (.well-known/openid-federation, /fetch) to retrieve current metadata and configurations of Federation participants, including Wallet Providers
CI_037	Issuance, Trust, Interoperability	Wallet Provider Trust Establishment	Credential Issuer establishes trust in the Wallet Provider as an authorized Federation entity by querying Federation API endpoints (e.g., .well-known/openid-federation, /fetch) and validating the provider's federation status through official channels and trust verification processes.
CI_038	Voided	Voided	Voided
CI_039	Voided	Voided	Voided
CI_040	Issuance, Security	Recommended PAR Response Request URI Validity Duration	request_uri validity time is set to less than one minute
CI_041	Issuance, Security	PAR response Request URI Cryptographic Random Value Integration	Generated request_uri includes a cryptographic random value of at least 128 bits
CI_042	Issuance, Security	Recommended PAR response Request URI Length Limitation	Complete request_uri doesn't exceed 512 ASCII characters

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_043	Issuance, Interoperability	PAR Response Successful Verification Response	When verification is successful, Credential Issuer returns an HTTP response with 201 status code
CI_044	Issuance, Interoperability	JSON PAR Response Structure	HTTP response message body uses application/json media type (**RFC 8259**) and includes the required top-level parameters
CI_044a	Issuance, Security	PAR Response request_uri Parameter	HTTP response includes request_uri parameter containing the generated one-time authorization URI
CI_044b	Issuance, Security	PAR Response expires_in Parameter	HTTP response includes expires_in parameter specifying the validity duration in seconds
CI_045	Issuance, Interoperability	PAR Response HTTP Status Code table	When PAR request processing encounters errors, the Credential Issuer responds as defined in **RFC 9126**, according to the *Table of HTTP Status Codes* (page 571).
CI_046	Issuance, Security and Privacy	User Identity Verification during authorization request	Authorization Server successfully verifies the identity of the User who owns the credential through appropriate authentication mechanisms, confirming user ownership before proceeding with credential authorization
CI_047	Issuance, Security	Request URI One-Time Use and Expiration in the Authorization Request	Credential Issuer treats each request_uri value as single-use only and successfully rejects any expired requests
CI_048	Issuance, Security	Optional Duplicate Request Tolerance in the Authorization Request	Credential Issuer allow duplicate requests when they result from User reloading or refreshing their user-agent (derived from **RFC 9126**)

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_049	Issuance, Security	PAR Request Identification in the Authorization Request	Credential Issuer successfully identifies and correlates each authorization request as a direct result of a previously submitted PAR (derived from **RFC 9126**)
CI_050	Issuance, Security	Request URI Parameter Requirement of the Authorization Request	Credential Issuer rejects all Authorization Requests that do not contain the request_uri parameter, since PAR is the exclusive method for passing Authorization Requests from the Wallet Instance (derived from **RFC 9126**).
CI_051	Issuance, Security	CieID High-Level Authentication for PID	PID Provider successfully performs User authentication based on CieID scheme with LoA High (CIE L3). This case applies to PID, not to IT-Wallet ID.
CI_052	Issuance, Security and Privacy	User Consent for PID Issuance	PID Provider obtains explicit User consent before proceeding with PID issuance
CI_053	Issuance, Privacy	Optional Contact Details Collection	PID Provider MAY request User's contact details (email, phone number) for sending notifications about the issued PID
CI_054	Presentation, Issuance Security	PID or IT-Wallet ID User Authentication	(Q)EAA Provider successfully performs User authentication as defined in credential-issuance-endpoint: User Authentication Method Selection
CI_055	Presentation, Issuance, Interoperability	OpenID4VP Protocol Usage	(Q)EAA Provider uses OpenID4VP according to credential-issuance-endpoint: User Authentication Method Selection
CI_056	Presentation, Issuance, Security	Presentation Request Delivery	(Q)EAA Provider successfully provides the presentation request to the Wallet

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_057	Issuance, Privacy	Optional Contact Details Collection for Credentials	Credential Issuers request User's contact details (email, phone number) for sending notifications about issued Digital Credential(s)
CI_058	Issuance, Interoperability	Authorization Response parameters validation	Credential Issuer successfully sends an authorization code response to the Wallet Instance that includes all three required parameters
CI_058a	Issuance, Security	Authorization Response code parameter validation	Authorization code response includes the authorization code parameter
CI_058b	Issuance, Security	Authorization Response state parameter validation	Authorization code response includes the state parameter matching the original request
CI_058c	Issuance, Security	Authorization Response iss parameter validation	Authorization code response includes the iss parameter identifying the issuer
CI_059	Issuance, Interoperability	Authorization Response HTTP Status Code table	When Authorization Response encounter errors, the Authorization Server redirects the User to the redirect_uri HTTP status code 302 according to the HTTP Status Code table
CI_060	Issuance, Security	Authorization Code Issuance Verification of the Token request	Credential Issuer ensures the Authorization code is issued to the authenticated Wallet Instance (**RFC 6749**) and has not been replayed
CI_061	Issuance, Security	Authorization Code Validity and Usage Check of the Token Request	Credential Issuer verifies the Authorization code is valid and has not been previously used (**RFC 6749**).
CI_061a	Issuance, Security	PKCE `code_verifier` Verification in Token Request	The Credential Issuer verifies that the `code_verifier` parameter is present and matches the `code_challenge` associated with the authorization code (**RFC 7636**); in case of mismatch, the request is rejected.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_062	Issuance, Security	Redirect URI Matching Validation of the Token Request	Credential Issuer confirms the redirect_uri exactly matches the value included in the previous Request Object (see Section 3.1.3.1. of [OIDC]).
CI_063	Issuance, Security	DPoP Proof JWT Validation of the Token Request	Credential Issuer successfully validates the DPoP Proof JWT, according to (**RFC 9449**) Section 4.3.
CI_064	Issuance, Interoperability	Access Token Provision in the token response	Credential Issuer provides the Wallet Instance with a valid Access Token upon successful authorization
CI_065	Issuance, Interoperability	Optional Refresh Token Provision	If supported by the Credential Issuer, a Refresh Token is provided to the Wallet Instance
CI_066	Issuance, Security	DPoP Key Binding for Access Token and Refresh Token	Both Access Token and Refresh Token (when issued) are cryptographically bound to the DPoP key
CI_067	Issuance, Interoperability	Token Response HTTP Status Code table	When any errors occur during the validation of the Token Request, the Authorization Server return an error response as defined in **RFC 6749**, according to the HTTP Status Code Table
CI_068	Issuance, Interoperability	c_nonce Provision	Credential Issuer provides a c_nonce value to the Wallet Instance
CI_069	Issuance, Security	C_nonce Format and Security	The c_nonce parameter is provided as a string value with sufficient unpredictability to prevent guessing attacks, serving as a cryptographic challenge that the Wallet Instance uses to create proof of possession of the key (proofs claim)

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_070	Issuance, Security	C_nonce Reusability and Renewal	The received c_nonce value can be reused by the Wallet Instance to create multiple proofs until the Credential Issuer provides a new c_nonce value
CI_071	Issuance, Interoperability	JWT Proof Required Claims Validation	JWT proof successfully includes all required claims as specified in the Token Request table
CI_072	Issuance, Security	Batch JWT Proof Required Claims Validation	Credential Issuer successfully verify that the jwk attribute in each key proof is unique
CI_073	Issuance, Interoperability	Credential Request Key Proof Type Declaration	Key proof is explicitly typed using appropriate header parameters defined for the respective proof type
CI_074	Issuance, Security	Asymmetric Algorithm Requirement in the Credential Request	The header parameter alg indicates a registered asymmetric digital signature algorithm and is never set to none
CI_075	Issuance, Security	Credential Request Public Key Signature Verification	Signature on the key proof is successfully verified using the public key specified in the header parameter
CI_076	Issuance, Security	Private Key Header Exclusion in the Credential Request	Header parameters do not contain any private key material
CI_077	Issuance, Security	c_nonce Matching in the Credential Request	When a c_nonce value was previously provided by the server, the nonce claim in the JWT exactly matches this c_nonce value
CI_078	Issuance, Security	JWT Temporal Validity in the Credential Request	The creation time of the JWT (via iat claim or server-managed timestamp through nonce claim) falls within the server's acceptable time window
CI_079	Voided	Voided	Voided
CI_080	Issuance, Interoperability	Recommended Fresh Cryptographic Key Generation in the Credential Request	Credential Issuer registers all issued Credentials in a revocation registry for potential future revocation needs

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_081	Issuance, Security	Recommended Deferred Flow support	When the requested Credential cannot be issued immediately and requires more time, the Credential Issuer supports the Deferred Flow
CI_081a	Issuance, Security	Deferred Flow batch issuance consistency	When using the Deferred Flow for batch issuance, the same transaction_id allows retrieval of all Credentials requested in the batch.
CI_082	Issuance, Security	DPoP JWT Proof and Access Token Validation in the Credential Response	Credential Issuer successfully validates the DPoP JWT Proof based on the steps defined in Section 4.3 of (**RFC 9449**) procedures and confirms the Access Token is valid and suitable for the requested Credent
CI_083	Issuance, Security	Key Material Proof of Possession Validation in the Credential Response	Credential Issuer validates the proof of possession for the key material to which the new Credential will be bound, according to OpenID4VCI Appendix F.4.
CI_084	Issuance, Security	Credential Creation and Binding in the Credential Response	When all validation checks succeed, Credential Issuer creates a new Credential cryptographically bound to the validated key material and provides it to the Wallet Instance
CI_084a	Issuance, Security	Credential type check	Credential Issuer ensure the credential type matches the request before issuing the new Credential
CI_085	Issuance, Interoperability	Credential Response Table of HTTP Status Codes	When the Credential Request does not contain a valid Access Token, the Credential Endpoint returns an error response such as defined in Section 3 of [**RFC 6750**], according to the *Table of HTTP Status Codes* (page 587).

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_086	Issuance, Interoperability	Unified Notification ID for Batch Operations	For batch-issued Digital Credentials, a single notification_id covers the entire batch-issued Credentials. The notification response applies to all Credentials, any partial failure is treated as a batch failure.
CI_087	Issuance, Interoperability	Notification Response Table of HTTP Status Codes	When the Notification Request does not contain a valid Access Token, the Notification Endpoint returns an error response such as defined in Section 3 of [**RFC 6750**], according to the *Table of HTTP Status Codes* (page 592).
CI_088	Issuance, Security	Access Token Scope Restriction	Access Token obtained through a Refresh Token flow is successfully limited to three specific endpoints: Deferred endpoint, Notification endpoint and Credential endpoint
CI_088a	Issuance, Security	Deferred Endpoint Access Authorization	Access Token allows access to Deferred endpoint for obtaining new Digital Credentials after interval or readiness notification
CI_088b	Issuance, Security	Notification Endpoint Access Authorization	Access Token allows access to Notification endpoint for notifying Digital Credential deletion to the Credential Issuer
CI_089c	Issuance, Security	Credential Endpoint Access Authorization	Access Token allows access to Credential endpoint for Digital Credential refresh/re-issuance of existing credentials
CI_090	Voided	Voided	Voided
CI_091	Issuance,Interoperability	OAuth Client Attestation PoP Validation for Refresh	Credential Issuer successfully validates the OAuth-Client-Attestation-PoP parameter based on Section 5 of [OAUTH-ATTESTATION-CLIENT-AUTH]

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_092	Issuance, Security	DPoP Proof JWT Validation for Refresh	Credential Issuer validates the DPoP Proof JWT according to (**RFC 6949**) Section 4.3.
CI_093	Issuance, Security	Refresh Token Validity and Key Binding Check	Credential Issuer verifies the Refresh Token is not expired, not revoked, and is bound to the same DPoP keys used in the DPoP Proof JWT
CI_094	Issuance, Security	Access Token Generation and Bound	When all validation checks succeed, Credential Issuer generates new Access Token and new Refresh Token, both bound to the DPoP key
CI_095	Issuance, Security	Successful Refresh Token Response	Both the Access Token and the Refresh Token are sent back to the Wallet Instance
CI_096	Issuance, Security	Invalid Refresh Token Error Handling	When the Refresh Token is expired or invalid, Credential Issuer issues an error response with error type member set to invalid_grant
CI_097	Issuance, Security	Refresh Token Confidentiality Protection	Confidentiality of Refresh Tokens is guaranteed both in transit and storage through appropriate encryption and secure handling mechanisms
CI_098	Issuance, Security	TLS-Protected Refresh Token Transmission	All token transmissions use TLS-protected connections, ensuring encrypted communication channels for token exchange
CI_099	Issuance, Security	Refresh Token Security Properties	Refresh tokens are generated with unguessable values and protected from modification through cryptographic integrity mechanisms
CI_100	Voided	Voided	Voided
CI_101	Issuance, Security	Consistent DPoP Key Binding between Refresh and Access token	Access Tokens and Refresh Tokens are bound to the same DPoP key
CI_102	Issuance, Security	DPoP Proof Requirement for Refresh Token	DPoP Proof is required for all refresh token operations to obtain new Access Tokens

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_103	Issuance, Security	Consistent DPoP Key Usage for Refresh Token	The same DPoP key is used to generate Access Token DPoP Proofs across all Credential Requests
CI_104	Issuance, Security	Refresh Token Usage Duration Management	Credential Issuers manage and limit the duration for which refresh tokens can be used to refresh credentials versus requiring complete re-issuance flow restart. As specified in OPENID4VC-HAIP
CI_105	Issuance, Security	Recommended Re-issued Credential Expiration Alignment	New Digital Credentials obtained through re-issuance flow maintain the same expiration as the refreshed credential
CI_106	Issuance, Security	Post-Expiration Issuance Requirement	Once a Digital Credential expires, Users complete the entire issuance process again to obtain new Digital Credentials
CI_107	Issuance, Security	Consistent Issuer Requirement for Re-Issuance	New Digital Credentials are issued exclusively by the same Credential Issuers that originally provided the existing credentials to the same Wallet Instance
CI_108	Issuance, Security	Refresh Token Scope Limitation for Re-issuance	Access Tokens obtained through Refresh Token flows are restricted from issuing Digital Credentials not already present in the Wallet Instance (first-time-issuance)
CI_109	Issuance, Security	Re-issuance Process Scope Limitation	The re-issuance process is limited to two specific update types: Data model/format technical updates and User's attribute set updates
CI_110	Issuance, Security	Not Recommended Technical Update User Interaction	For data model/format technical updates, the replacement and storage of Digital Credentials don't require direct user involvement

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_111	Issuance, Security and Privacy	Attribute Update User Authorization	For User's attribute set updates, the Wallet Instance informs the User about attribute data set changes and requests explicit User authorization before storing the new Digital Credential
CI_112	Issuance, Security	Expiry Date Consistency for Re-Issuance	Newly issued Digital Credentials maintain the same expiry date as the previous credential version
CI_113	Issuance, Privacy	Optional Out-of-Band Re-Issuance Update Notification	When a Digital Credential requires updating, Credential Issuers send notifications to Users through registered out-of-band communication channels (email, SMS, push notifications)
CI_114	Issuance, Security	First-Time Issuance Restriction for Refresh Tokens	Access Tokens obtained through Refresh Token flows are prohibited from being used for first-time issuance of Digital Credentials
CI_115	Voided	Voided	Voided
CI_116	Issuance, Privacy	User Consent for Attribute-Based Re-issuance	For re-issuance processes triggered by attribute changes, User consent is obtained before storing the new Digital Credential
CI_117	Data Model and lifecycle, Interoperability	Italian PID User attributes	The Italian PID is successfully provided with the User attributes defined in the *PID parameters table* (page 382).
CI_118	Data Model and lifecycle, Issuance, Interoperability	(Q)EAA Credential Formats	(Q)EAA are Issued to a Wallet Instance in SD-JWT VC or mdoc-CBOR data format.
CI_119	Voided	Voided	Voided
CI_120	Data Model and lifecycle, Security	Signature of the SD-JWT credential	The Credential in SD-JWT format is signed using the Issuer's private key.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_121	Data Model and lifecycle, Interoperability	SD-JWT Type Metadata Provision	SD-JWT is successfully provided with complete Type Metadata for the issued Digital Credential, conforming to Sections 6 and 6.3 of SD-JWT-VC specifications.
CI_122	Data Model and lifecycle, Security	SD-JWT Payload Structure Validation	SD-JWT payload contains the required *_sd_alg* claim and all other mandatory claims as specified, with proper formatting and structure.
CI_123	Data Model and lifecycle, Security	SD-JWT Hash Algorithm Declaration	The *_sd_alg* claim is present and set to a supported algorithm defined in Section Cryptographic Algorithms, indicating the hash algorithm used by the Issuer to generate digests as described in SD-JWT Section 4.1.1.
CI_124	Data Model and lifecycle, Interoperability	SD-JWT Selective Disclosure Organization	Non-selectively disclosable claims appear directly in the SD-JWT payload as they are, while selectively disclosable claim digests (plus any decoys) are properly organized within *_sd* arrays as described in Section 4.2.4.1.
CI_125	Data Model and lifecycle, Interoperability	SD-JWT Disclosure Integrity Verification	Each digest value successfully validates against its corresponding disclosure, with disclosures containing the required components: random salt, claim name (for object elements), and claim value.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_126	Data Model and lifecycle, Interoperability	Recommendation on SD-JWT Nested Object Selective Disclosure	In a nested SD-JWT payload, every claim at each level of the JSON structure is explicitly marked as either selectively disclosable or not. As a consequence, the `_sd` claim containing digests can legitimately appear multiple times at different levels within the SD-JWT.
CI_127	Data Model and lifecycle, Interoperability	SD-JWT Disclosure digests positioning	Array element disclosures are correctly positioned, with digests and decoy digests replacing original claim values in their exact array positions, maintaining structural integrity as specified in Section 4.2.4.2 of SD-JWT.
CI_128	Data Model and lifecycle, Security	SD-JWT Array Element Digest Calculation	Array elements digest values are calculated using a hash function over the disclosures, containing: a random salt and the array element.
CI_129	Data Model and lifecycle, Privacy	SD-JWT Array Disclosures	Array-level selective disclosure operates correctly, allowing Holders to selectively disclose entire arrays or individual entries within arrays, as defined in Section 4.2.6 of SD-JWT.
CI_130	Data Model and lifecycle, Interoperability	SD-JWT Combined Format Disclosure Delivery	The Disclosures are successfully provided to the Holder together with the SD-JWT in the Combined Format for Issuance, formatted as an ordered series of base64url-encoded values with each value properly separated by a single tilde ('~') character.
CI_131	Data Model and lifecycle, Interoperability	SD-JWT JOSE Header Parameter	The JOSE header contains the parameter in the Credential *SD-JWT Parameters Table* (page 359).

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_132	Data Model and lifecycle, Interoperability	SD-JWT Payload Claims	The JWT payload contains claims in the Credential *SD-JWT Parameters Table* (page 359).
CI_133	Data Model and lifecycle, Interoperability	SD-JWT Status List Parameter Structure	When status parameter is set to status_list, it is a JSON Object containing the following sub-parameters: *idx* and *uri*.
CI_134	Data Model and lifecycle, Interoperability	SD-JWT Optional Credential Type Metadata Retrieval	The Credential Type Metadata JSON Document is successfully retrieved directly from the *well-known* endpoint.
CI_134a	Data Model and lifecycle, Interoperability	SD-JWT Case-Insensitive URI Matching	When retrieving Credential Type Metadata via vct, URN string literal matching is performed in a case-sensitive manner, while the system operates without requiring the .well-known endpoint (as specified in Section 6.3.3 of SD-JWT-VC), maintaining compatibility options for implementers who choose to use it for interoperability with other systems.
CI_135	Data Model and lifecycle, Interoperability	Metadata Document JSON Object Structure	The Type Metadata document, if provided, is a JSON object containing the parameters defined in Section 6.2 of SD-JWT-VC.
CI_136	Data Model and lifecycle, Interoperability	Additional PID Claims	Depending on the Digital Credential type specified in vct, additional claims data is successfully incorporated when required
CI_137	Data Model and lifecycle, Interoperability	Mdoc Credential Format	The mdoc data elements are successfully encoded in CBOR format according to **RFC 8949** specifications, ensuring proper binary serialization and compatibility with ISO/IEC 18013-5 standard.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_138	Data Model and lifecycle, Interoperability	Mdoc Component Structure Organization	The mdoc Digital Credential is properly structured into distinct components including namespaces (*nameSpaces*) and cryptographic proof (*issuerAuth*)
CI_139	Data Model and lifecycle, Privacy	Mdoc Mobile Security Object Integrity Verification	The Mobile Security Object (MSO) correctly stores cryptographic digests of attributes within nameSpaces, enabling Relying Parties to validate disclosed attributes against corresponding digestID values while maintaining privacy of undisclosed information. See *Mobile Security Object* (page 367) for details
CI_140	Data Model and lifecycle, Interoperability	Mdoc-CBOR Structure Compliance	The mdoc-CBOR Digital Credential successfully conforms to the required structure as specified in the compliance *table* (page 366).
CI_141	Data Model and lifecycle, Interoperability	Namespace Structure Organization	The nameSpaces correctly contains one or more nameSpace entries, each properly identified by a unique name for organized data categorization.
CI_142	Data Model and lifecycle, Interoperability	Namespaces IssuerSignedItemBytes Encoding	Within each nameSpace, one or more IssuerSignedItemBytes are correctly encoded as CBOR byte strings with Tag 24 (#6.24(bstr .cbor)), appearing as 24(«... ») in diagnostic notation.
CI_143	Data Model and lifecycle, Interoperability	Namespace Disclosure Information Attributes	Each IssuerSignedItemBytes successfully represents the disclosure information for corresponding digests within the *Mobile Security Object* and contain all the attributes as specified in the compliance *table* (page 370).

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_144	Data Model and lifecycle, Interoperability	Namespace ElementIdentifier Inclusion	All elementIdentifiers in the elementIdentifiers attribute *table* (page 370) are properly included in the Digital Credential encoded in mdoc-CBOR within their respective nameSpaces, unless, otherwise specified.
CI_145	Data Model and lifecycle, Interoperability	Mobile Security Object COSE Structure	The issuerAuth successfully represents the Mobile Security Object as a properly formatted COSE Sign1 Document according to **RFC 9052**, containing the complete required data structure with: protected header, unprotected header, payload and signature components.
CI_146	Data Model and lifecycle, Interoperability	Mobile Security Object Protected Header Parameter Encoding	The protected header successfully contains the parameters properly encoded in CBOR format according to the corresponding *table* (page 367).
CI_147	Data Model and lifecycle, Interoperability	Mobile Security Object Protected Header Algorithm	The protected header successfully contains the required signature algorithm parameter.
CI_147a	Data Model and lifecycle, Interoperability	Not Recommended elements in the Mobile Security Object Protected Header	The protected header does not contain elements different from the signature algorithm
CI_148	Data Model and lifecycle, Interoperability	Mobile Security Object Unprotected Header Parameter Encoding	Unless otherwise specified, the unprotected header contain the parameters according to the corresponding *table* (page 367).
CI_148a	Data Model and lifecycle, Interoperability	Mobile Security Object x5chain inclusion	The *x5chain* is correctly included in the unprotected header

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_149	Data Model and lifecycle, Interoperability	Mobile Security Object Payload Structure	The payload successfully contains the MobileSecurityObject properly encoded as a byte string (bstr) using CBOR Tag 24, with the content-type COSE Sign header parameter correctly excluded from the structure.
CI_150	Data Model and lifecycle, Interoperability	Mobile Security Object Attributes Compliance	The MobileSecurityObject successfully contains all required attributes as specified in the compliance *table* (page 368), with proper formatting and values unless explicitly exempted by specification requirements.
CI_151	Issuance, Interoperability	Wallet Instance State verification	Credential Issuer successfully verifies that Wallet Instance is in Operational or Valid state and proceeds with Digital Credential issuance.
CI_152	Data Model and lifecycle, Interoperability	Credential State Management for Activation	A Digital Credential successfully transitions to Valid state when start date of validity is reached, or when unsuspension process is completed for a previously suspended (Q)EAA.
CI_153	Data Model and lifecycle, Interoperability	Automatic Credential Expiration Management	A Digital Credential successfully transitions to the Expired state when it automatically expires upon reaching its end date of validity (PID/(Q)EAA EXP),
CI_154	Data Model and lifecycle, Interoperability	Credential Revocation Process Management	A Digital Credential successfully changes from Issued, Valid or Suspended states to Revoked state when it is actively revoked by the Credential Issuer by a revocation process (PID/(Q)EAA REV).

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_155	Data Model and lifecycle, Interoperability	Digital Credential Revocation Use Cases	Digital Credential Revocation is correctly implemented for every use case
CI_155a	Data Model and lifecycle, Interoperability	Digital Credential Revocation - Technical Security Compromise	Digital Credential is successfully revoked when cryptographic material compromise is detected, with immediate invalidation and status update in revocation registries.
CI_155b	Data Model and lifecycle, Interoperability	Digital Credential Revocation - User Request	Digital Credential is successfully revoked upon explicit User request
CI_155c	Data Model and lifecycle, Interoperability	Digital Credential Revocation - Attribute Updates	Digital Credential is automatically revoked when Authentic Sources notify attribute changes that invalidate the Credential, triggering reissuance process if applicable.
CI_155d	Data Model and lifecycle, Interoperability	Digital Credential Revocation - Source-Initiated Revocation	Digital Credential is immediately revoked when Authentic Source notifies revocation of underlying attributes
CI_155e	Data Model and lifecycle, Interoperability	Digital Credential Revocation - Wallet Instance Revocation	All Digital Credentials are revoked when the containing Wallet Instance is revoked
CI_155f	Data Model and lifecycle, Interoperability	Digital Credential Revocation - Judicial/Supervisory Order	Digital Credential is immediately revoked when illegal activities are reported by Judicial or Supervisory Bodies
CI_155g	Data Model and lifecycle, Interoperability	PID Revocation - Identity Provider Breach	PID is revoked when breach detection occurs in the Identity Provider used for User authentication during PID issuance
CI_156	Data Model and lifecycle, Interoperability	Optional (Q)EAA Revocation Following PID Revocation	(Q)EAA Provider successfully evaluates PID revocation status and revoke associated (Q)EAA credentials

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_157	Data Model and lifecycle, Interoperability	(Q)EAA State Change to Suspended	(Q)EAA successfully transitions from Issued or Valid state to Suspended state when it is suspended by the Credential Issuer ((Q)EAA SUSP)
CI_158	Data Model and lifecycle, Interoperability	(Q)EAA Suspension Duration and State Recovery	Suspended (Q)EAA remains in Suspended state until suspension conditions are resolved and credential is restored to previous state Issued or Valid, or transitions to Revoked, Expired, or deleted
CI_158a	Data Model and lifecycle, Interoperability	(Q)EAA Suspension - Attribute Validity Changes	The suspension of a (Q)EAA is based on the validity status of its contained attributes
CI_158b	Data Model and lifecycle, Interoperability	(Q)EAA Suspension - User Request	(Q)EAA successfully transitions to Suspended state upon explicit User request.
CI_159	Data Model and lifecycle, Interoperability	Batch-Issued Digital Credentials Individual Lifecycle Management	Each Digital Credential from a batch issuance successfully enters its own independent lifecycle state machine. All state transitions (Issued → Valid → Expired/Suspended/ Revoked) still occur on a per Credential basis, using Credential's individual parameters
CI_160	Data Model and lifecycle, Interoperability	Digital Credential Generation and Storage	Credential Issuer successfully generates Digital Credential following issuance request completion and stores it in local storage immediately after successful issuance.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_160a	Data Model and lifecycle, Interoperability	Digital Credential State Management	Credential Issuer successfully updates Digital Credential state locally for revocation, suspension, or unsuspension events triggered by technical security reasons, User requests, or by external entities (e.g., Users and Authentic Sources)
CI_160b	Data Model and lifecycle, Interoperability	Digital Credential Data Purging	Credential Issuer successfully removes Digital Credentials from local storage after reaching the Expired state or based on Credential Issuer retention policies.
CI_161	Data Model and lifecycle, Interoperability	Credential Issuer Direct Validity Status Management	Credential Issuer directly manages the validity status of issued Digital Credentials and correctly manages the triggering of the Digital Credential revocation or suspension process by other actors.
CI_161a	Data Model and lifecycle, Interoperability	User-Initiated Revocation via Issuer Web Service	User successfully initiates Digital Credential revocation/suspension through Credential Issuer's web service, with authentication verification and revocation request processing.
CI_161b	Data Model and lifecycle, Interoperability	Authentic Source-Triggered Revocation	Authentic Source successfully triggers Digital Credential revocation/suspension process when Credential attributes are updated or change validity status.
CI_162	Data Model and lifecycle, Security	User Access to Issuer Web Portal with LoA Authentication	Credential Issuer ensures that users can successfully access the secure area of its web portal. This access requires authentication with a Level of Assurance that is at least equal to the Level of Assurance used during the initial credential issuance.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_162a	Data Model and lifecycle, Interoperability	User Access to Digital Credentials Database View	Credential Issuer successfully provides Users with complete view of all their Digital Credentials contained in the Issuer's database through the web portal
CI_162b	Data Model and lifecycle, Interoperability	User Data Authenticity Verification via Web Portal	Credential Issuer successfully enables Users to verify data authenticity of their Digital Credentials through the web portal
CI_162c	Data Model and lifecycle, Interoperability	User Validity Status Management via Web Portal	Credential Issuer successfully allows Users to view and update validity status of their Digital Credentials through the web portal (revoke their Digital Credentials and, if it is supported by the Issuer, suspend them).
CI_163	Data Model and lifecycle, Interoperability	PID Revocation and Wallet Instance Status Update on New PID Issuance	When User activates another Wallet Instance from the same Wallet Provider using the same Wallet Solution and obtains a new PID, the previous PID is successfully revoked and the previous Wallet Instance successfully transitions to operational status, ensuring single active PID per User per Wallet Provider.
CI_164	Data Model and lifecycle, Interoperability	PID Revocation Following User Death and ANPR Status Change	User's death successfully triggers change in validity status of User's identification attributes in the public registry (ANPR), which automatically produces PID revocation.
CI_165	Data Model and lifecycle, Interoperability	PID Revocation Notification Requirement	Credential Issuer successfully sends notification of PID revocation event to the User within 24 hours of revocation occurrence

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_166	Data Model and lifecycle, Interoperability	Recommended Non-PID Credential Revocation Notification	Credential Issuer sends notification of revocation event to the User for any Digital Credential other than PID
CI_167	Data Model and lifecycle, Interoperability	Multi-Channel Revocation Notification Delivery	Credential Issuer successfully delivers revocation notifications through verified and secure communication channels (email, telephone, or other authenticated channels), including complete information about the Credential revocation status and reason.
CI_168	Data Model and lifecycle, Interoperability	Digital Credential Status Update on Revocation	Credential Issuer successfully updates Digital Credential status immediately when revocation occurs
CI_169	Data Model and lifecycle, Interoperability	Wallet Instance Status Monitoring for the Digital Credential Status Update	Credential Issuer establishes a monitoring mechanism of the current statuses of all the Key Attestations related to the Wallet Instances to which the Credentials were issued.
CI_170	Data Model and lifecycle, Interoperability	Credential Status Update Following Data Change Notification	Credential Issuer successfully updates Credential Status according to the validity mechanism's defined mode upon receiving notification from Authentic Source
CI_170a	Data Model and lifecycle, Interoperability	User Notification of Credential Update	Credential Issuer successfully notifies User of credential changes through registered out-of-band communication channel
CI_171	Data Model and lifecycle, Interoperability	User Notification on Credential Status INVALID	Credential Issuer successfully informs User when Credential Status changes to INVALID

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_172	Data Model and lifecycle, Interoperability	Batch Status Update Processing as Individual Changes	Credential Issuer successfully handles single batch status update request (referencing batch notification_id) from authorized entities (e.g. Wallet Instance via Notification Endpoint with event=credential_deleted, or Wallet Provider via PDND) as N separate individual status changes, with each Credential's status updated independently (for example, flipping its status-list bit to INVALID or SUSPENDED).
CI_173	Data Model and lifecycle, Interoperability	Batch revocation upon credential Update Request	Credential Issuer successfully processes batch update request as revoke all requests, marking every Credential in the batch as revoked and emitting single notification covering the entire batch.
CI_174	Data Model and lifecycle, Interoperability	Batch Credential Deletion	User-driven deletion successfully removes the entire batch as Wallet UI surfaces a batch as one Credential, with deletion request using the batch's notification_id applying to all Credentials in that batch, as it is not possible to delete or revoke just one Credential.
CI_175	Data Model and lifecycle, Interoperability	OAuth Status List Support for Long-Lived Digital Credentials	OAuth Status List (TOKEN-STATUS-LIST) is successfully supported for verification of validity status of long-lived Digital Credentials in both remote and proximity scenarios.
CI_176	Data Model and lifecycle, Interoperability	Digital Credential Index Allocation and Status Mapping	Each Digital Credential is successfully allocated an index during issuance that represents its position within the bit array, with the value of the bit(s) at this index corresponding to the Digital Credential's status.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_177	Data Model and lifecycle, Interoperability	Status List Token Cryptographic Format	Status List is successfully provided within cryptographically signed Status List Token in JWT format.
CI_178	Data Model and lifecycle, Interoperability	Status List Bit Configuration for Digital Credentials	Credential Issuer successfully defines number of bits k (either 1, 2, 4, 8) that represents the amount of bits used to describe the status of each Digital Credential within Status List, with Credential Issuer configuring the number of bits and each Credential having 2^k possible states.
CI_179	Data Model and lifecycle, Interoperability	Status List Byte Array Creation and Credential Position Assignment	Credential Issuer successfully creates byte array of size = (amount of Digital Credentials) * k / 8 or greater, with each byte corresponding to 8/k statuses depending on k value (8 if k=1, 4 if k=2, 2 if k=4, or 1 if k=8), and assigns each issued Digital Credential to a position in the array.
CI_180	Data Model and lifecycle, Interoperability	Status List Digital Credential Status Values Setting in Byte Array	Credential Issuer successfully sets status values for all issued Digital Credentials within the byte array, with each Digital Credential status identified using an index that maps to specific bits within the byte array, index counting from 0 to (amount of Digital Credential) - 1, bits counted from least significant bit (”0”) to most significant bit (”7”), and all bits of the byte array at a particular index set to a status value
CI_181	Data Model and lifecycle, Interoperability	Status List Digital Credential Byte Array Compression	Credential Issuer successfully compresses the byte array using DEFLATE [**RFC 1951**] with the ZLIB [**RFC 1950**] data format

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_181a	Data Model and lifecycle, Interoperability	Status List Digital Credential Recommended Compression Level	Implementations successfully use the highest compression level available for byte array compression.
CI_182	Data Model and lifecycle, Interoperability	Status List Endpoint Availability	Credential Issuer successfully makes available to Relying Parties and Wallet Instances an endpoint to request Status Lists.
CI_183	Data Model and lifecycle, Interoperability	Status List Digital Credential Status Values Definition	Credential Issuer successfully uses the values for possible Statuses defined in the *Token Status Lists* (page 411) section.
CI_184	Data Model and lifecycle, Interoperability	Status List Optional Digital Credential Status States Definition	Credential Issuer successfully adds other states besides those described above when choosing the number of bits for conveying statuses of issued Digital Credentials, with careful consideration for information disclosure to Relying Parties when adding many different states for Digital Credential lifecycle.
CI_185	Data Model and lifecycle, Interoperability	Status List Token Parameters at Endpoint	Status List Token is successfully available at the Status List Endpoint and contains the parameters in the corresponding *table* (page 413)
CI_186	Data Model and lifecycle, Interoperability	Recommended Status List Token Short-Lived Expiration Setting	Credential Issuer successfully sets exp claim so that Status List Token is short-lived, typically with exp claim not exceeding iat claim by more than 24 hours.
CI_187	Data Model and lifecycle, Interoperability	JSON-Encoded Status List Structure	The JSON-encoded Status List's structure correctly conforms to the corresponding *table* (page 414)

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_188	Data Model and lifecycle, Interoperability	Status List Digital Credential Local Storage	Credential Issuers successfully store generated Digital Credential locally with minimum set of data required to manage its lifecycle, including the validity status of that Digital Credential.
CI_189	Data Model and lifecycle, Interoperability	Digital Credential Status List Claim Inclusion	Credential Issuers successfully include *status_list* claim within the JSON Object value of the status claim of the Digital Credential once generated.
CI_190	Data Model and lifecycle, Interoperability	Status List Claim JSON Object Parameters	The value of *status_list* claim is successfully a JSON Object with the corresponding *parameters* (page 415).
CI_191	Data Model and lifecycle, Interoperability	Status List Endpoint Successful Response	Status List Endpoint successfully responds with Status List Token using HTTP status code in the 2xx range, with Status Provider using content-type application/statuslist+jwt for Status List Token in JWT format in the successful response.
CI_192	Data Model and lifecycle, Interoperability	HTTP Status List Response Gzip Content-Encoding	HTTP response successfully uses gzip Content-Encoding as defined in [**RFC 9110**].
CI_193	Data Model and lifecycle, Interoperability	Digital Credential Issuer Authorization and Registration	Credential Issuer successfully registers its own Digital Credentials in the catalogue.
CI_194	Data Model and lifecycle, Interoperability	Digital Credential Catalogue Required Information	Credential Issuer provides its credentials in the catalogue, along with the information in the corresponding *table* (page 194).
CI_195	Data Model and lifecycle, Interoperability	Credentials Array Element Information	Each element of the *Credentials* array correctly contains all the information defined in the First-level Fields *table* (page 195).

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_196	Data Model and lifecycle, Interoperability	IT-Wallet ID User attributes	The IT-Wallet ID is issued with the User attributes defined in the *IT-Wallet ID data model* (page 392), distinct from the PID attribute set.
CI_197	Issuance, Authentication	IT-Wallet ID primary authentication	The EAA Provider of IT-Wallet ID authenticates the User with CieID LoA High (CIE L3) as the primary method.
CI_198	Issuance, Authentication	eID Substantial Authentication with MRTD Verification	When CieID PIN is not used, the EAA Provider of IT-Wallet ID supports the eID Substantial Authentication with MRTD Verification flow in *eID Substantial Authentication with MRTD Verification for IT-Wallet ID Issuance* (page 457). Actors in that flow retain logs according to the General Log Retention Policies.
CI_199	Issuance, Authentication	L2+ Authorization Details	For IT-Wallet ID issuance with MRTD Verification, the PAR `authorization_details` array contains an object with `type` `it_l2+document_proof` and the required claims. If that object is absent, authentication falls back to CieID LoA High.
CI_200	Issuance, Authentication, Security	IdP–MRTD identity correlation	The EAA Provider binds the identity verified by the IdP (CieID/SPID Substantial) with the identity in the MRTD by comparing at least the tax identification number or an equivalent unique identifier present in both sources. A mismatch aborts issuance. LoA High primary authentication skips MRTD PoP.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_201	Presentation, Security	Identity matching with unique national identifier	When `personal_administrative_number` or `tax_id_code` is presented, the Relying Party uses it as the primary identity matching key before any reconciliation with a stored User record, as specified in *Identity Matching and Identity Reconciliation* (page 538).
CI_202	Presentation, Security	Attribute-Based Binding without unique identifier	When a PID is presented without `personal_administrative_number`, Attribute-Based Binding is not limited to a single pair of names. The Relying Party matches at least `family_name`, `given_name`, `birth_date` (or `birthdate`) and `place_of_birth` after normalisation, and MUST NOT complete matching on a subset that would allow person swap, as specified in *Identity Matching and Identity Reconciliation* (page 538). Session-based, issuer-attested or cryptographic binding MAY supplement the comparison.
CI_203	Issuance, Privacy	Batch issuance applicability	If the Credential Issuer metadata contains `batch_credential_issuance`, batch issuance applies only to the Credential types for which that metadata is advertised. The Issuer documents whether PID and IT-Wallet ID are in scope. A type without that metadata is issued as a single Credential.
CI_204	Issuance, Interoperability	WIA at issuance	At issuance the Credential Issuer validates a Wallet Instance Attestation (WIA), not a legacy «Wallet Attestation». WUA, where used, is a distinct artefact.

continues on next page

Table 20.33 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
CI_205	Issuance, Authentication	LoA nomenclature mapping	In the L2+ flow, LoA3 (ISO/IEC 29115) maps to eIDAS Substantial and to the national L2 means (SPID L2 / Cie ID Substantial). LoA High maps to eIDAS High and CIE L3. Tests fail if a Substantial authentication is recorded as High.

Credential Verifier Test Matrix

This section provides the set of test cases designed for technical implementers and development teams responsible for creating and deploying Credential Verifiers / Relying Parties.

Remote Credential Verifier Test Matrix

This section provides the set of test cases designed for technical implementers and development teams responsible for creating and deploying Credential Verifiers solutions for remote flows. It is also intended for assessment bodies inspecting and validating the implementations of Credential Verifiers solutions for remote flows.

Note

References about official OpenID4VP test plans will update this section in future releases.

Test Case ID	Purpose	Description	Expected Result
RPR-01	Same Device Flow	Verify HTTP redirect (302) URL.	Relying Party issues a correct URL using the base url provided within its metadata.
RPR-02	Cross Device Flow	Verify QR Code generation for Wallet Instance.	Relying Party issues QR Code successfully.
RPR-03	Cross Device Flow	Verify QR Code contains correct URL parameters.	Relying Party issues the QR-Code containing an URL using the base url provided within its metadata.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-04	Cross Device Flow	Test QR Code scanning in low light.	QR Code is scanned successfully.
RPR-05	Cross Device Flow	Verify QR Code error correction level.	QR Code remains readable if damaged.
RPR-06	Cross Device Flow	Test QR Code scanning with different devices.	QR Code is scanned successfully.
RPR-07	Request URI Method	Test `request_uri_method` as `post`.	Relying Party accepts Wallet Instance metadata via POST and replies with an updated Request Object.
RPR-08	Request URI Method	Test `request_uri_method` as `get`.	Relying Party issues the Request Object via HTTP GET response.
RPR-09	Request URI Method	Test absence of `request_uri_method`.	Relying Party accepts defaults to GET method.
RPR-10	Metadata	Verify parameters match Open ID Credential Verifier metadata.	Only allowed parameters will be considered.
RPR-11	User Consent	Test eligibility of a Credential Verifier in requesting User attributes.	User can modify data selection about optional attributes.
RPR-12	Authorization Response	Test sending of Presentation Response.	Relying Party receives and validates response with `state` and `nonce` values.
RPR-13	Authorization Response	Verify response encryption.	Relying Party evaluates the encrypted response using its public key (one of its keys).
RPR-14	Error Handling	Test invalid Request Object handling.	Error Response is sent.
RPR-15	Error Handling	Verify error logging.	Errors are logged appropriately.
RPR-16	Error Handling	Test recovery from Authorization Request Error.	Relying Party prompts the User to retry or scan new QR code.
RPR-17	Error Handling	Test fake HTTP Cookie.	Relying Party checks the User session consistency coupling the session http cookie and the state and nonces provided.
RPR-19	Redirect URI	Test redirection to Relying Party's endpoint.	User is redirected correctly, the endpoint works.
RPR-20	Redirect URI	Verify handling of invalid *redirect_uri*.	Error response is returned.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-23	Credential Presentation	Verify response format compliance.	Relying Party supports all the Credential format included within its `vp_formats_supported` metadata parameter.
RPR-24	Authorization Response	Test handling of response timeouts.	Retries must be successful unless response is acquired.
RPR-25	Error Handling	Verify handling of malformed claims in presentation payload.	Authorization Error Response is sent.
RPR-26	Error Handling	Verify handling of malformed claims in presented credentials.	Authorization Error Response is sent.
RPR-27	Error Handling	Test handling of expired requests.	Holder is notified of expiration.
RPR-28	Relying Party Response	Verify inclusion of response code.	Response code is cryptographically random.
RPR-29	Relying Party Response	Test handling of invalid response codes.	Error response is returned.
RPR-30	Status Endpoint	Verify handling of unauthorized access.	Unauthorized access is denied.
RPR-31	Status Endpoint	Test handling of invalid session IDs.	Error response is returned.
RPR-32	Redirect URI	Verify handling of expired sessions.	Error response is returned.
RPR-33	Redirect URI	Test handling of server errors.	Error response is returned.
RPR-34	Same and Cross Device Flow	Verify handling of slow network conditions.	Relying Party provides the http response within the maximum limit of 2 seconds.
RPR-36	Presentation Response	Verify handling of large response payloads.	Response is evaluated within appropriate security thresholds.
RPR-37	Presentation Response	Test handling of response encryption failures.	Error response is returned.
RPR-38	Error Handling	Verify handling of invalid signatures.	Authorization Error Response is sent.
RPR-39	Error Handling	Test handling of invalid nonce values.	Error response is returned.
RPR-40	Relying Party Response	Verify handling of malformed responses.	Error response is returned.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-41	Relying Party Response	Test handling of missing response parameters.	Error response is returned.
RPR-42	Status Endpoint	Verify handling of session timeouts.	Error response is returned.
RPR-43	Status Endpoint	Test handling of invalid status codes.	Error response is returned.
RPR-44	Redirect URI	Verify handling of invalid user sessions.	Error response is returned.
RPR-45	Redirect URI	Test handling of unavailable services.	Error response is returned.
RPR-46	Same Device Flow	Verify handling of user cancellations.	User can cancel the process.
RPR-47	Cross Device Flow	Test QR Code scanning with different apps.	QR Code is scanned successfully.
RPR-48	Cross Device Flow	Verify QR Code scanning with different lighting.	QR Code is scanned successfully.
RPR-49	Request URI Method	Test handling of unsupported content types.	Error response is returned.
RPR-50	User Consent	Verify user notification of consent changes.	User is informed about consent changes.
RPR-51	User Consent	Test user consent for sensitive data.	User can consent to sensitive data.
RPR-52	Authorization Response	Verify handling of response decryption failures.	Error response is returned.
RPR-53	Authorization Response	Test handling of response integrity checks.	Response integrity is verified.
RPR-54	Relying Party Response	Verify handling of response validation failures.	Error response is returned.
RPR-55	Relying Party Response	Test handling of response processing errors.	Error response is returned.
RPR-56	Protected Resource Endpoint	Verify handling of unauthorized session access.	Unauthorized access is denied.
RPR-57	Redirect URI	Verify handling of invalid redirect parameters.	Error response is returned.
RPR-58	Redirect URI	Test handling of redirect failures.	Error response is returned.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-59	Same Device Flow	Verify handling of user interruptions.	User can resume or cancel the process.
RPR-60	Request URI Method	Test handling of invalid HTTP methods.	Error response is returned.
RPR-61	User Consent	Verify user notification of consent revocation.	User is informed about consent revocation.
RPR-62	User Consent	Test user consent for optional data.	User can consent to optional data.
RPR-63	Authorization Response	Verify handling of response signature failures.	Error response is returned.
RPR-64	Authorization Response	Test handling of response format errors.	Error response is returned.
RPR-65	Error Handling	Verify handling of invalid JWT signatures.	Authorization Error Response is sent.
RPR-66	Error Handling	Test handling of invalid JWT claims.	Error response is returned.
RPR-67	Relying Party Response	Verify handling of response parsing errors.	Error response is returned.
RPR-68	Relying Party Response	Test handling of response timeout errors.	Error response is returned.
RPR-69	Status Endpoint	Verify handling of session expiration.	Error response is returned.
RPR-70	Status Endpoint	Test handling of session renewal errors.	Error response is returned.
RPR-71	Redirect URI	Verify handling of redirect loop errors.	Error response is returned.
RPR-72	Redirect URI	Test handling of redirect security errors.	Error response is returned.
RPR-73	Same Device Flow	Verify handling of user timeouts.	User is notified of timeout.
RPR-74	Cross Device Flow	Test QR Code scanning with different devices.	QR Code is scanned successfully.
RPR-75	Cross Device Flow	Verify QR Code scanning with different apps.	QR Code is scanned successfully.
RPR-76	Request URI Method	Test handling of unsupported HTTP methods.	Error response is returned.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-77	QR Code Generation	Verify that QR Code error correction level is Q (Quartile - up to 25%).	QR Code uses the required Q error correction level.
RPR-78	Wallet Instance Attestation (WIA) Request	Test that Wallet Instance Attestation (WIA) request uses standard DCQL query.	Wallet Instance Attestation (WIA) request correctly uses standard DCQL query.
RPR-79	Wallet Instance Attestation (WIA) Request	Verify that `claims` parameter is not included in DCQL query for Wallet Instance Attestation (WIA).	`claims` parameter is not included in DCQL query for Wallet Instance Attestation (WIA).
RPR-80	Wallet Instance Attestation (WIA) Request	Test that `vct_values` parameter is required in DCQL query for Wallet Instance Attestation (WIA).	`vct_values` parameter is correctly required in DCQL query.
RPR-81	Wallet Nonce	Test that Relying Party checks `wallet_nonce` when present.	Relying Party correctly checks `wallet_nonce` when present.
RPR-82	Response Types	Verify that `response_types_supported` is set to `vp_token` when present.	`response_types_supported` is correctly set to `vp_token`.
RPR-83	Redirect URI	Test that Relying Party correctly provides `redirect_uri` parameter to Wallet Instance.	Relying Party correctly provides and handles `redirect_uri`.
RPR-84	Flow Support	Test that Relying Party supports required remote flows.	Relying Party supports both Same Device and Cross Device flows.
RPR-85	Endpoint Security	Test that `request_uri` is attested by trusted third party.	`request_uri` parameter is properly attested by trusted third party.
RPR-86	Privacy Protection	Test that Relying Party correctly validates Wallet Instance metadata without User information.	Relying Party correctly evaluates Wallet Instance technical capabilities.
RPR-87	Request URI POST Method	Test that Relying Party supports receiving Wallet Instance metadata via POST to `request_uri` endpoint with content type `application/x-www-form-urlencoded`.	Relying Party correctly accepts and processes Wallet Instance metadata sent via POST with the required content type.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-88	Algorithm Validation	Test that JWT algorithm is supported and not `none` or MAC.	JWT algorithm is from supported list and not `none` or MAC identifier.
RPR-89	Media Type Validation	Test that JWT typ is set to `oauth-authz-req+jwt`.	JWT typ parameter is correctly set to `oauth-authz-req+jwt`.
RPR-90	Response Mode Validation	Test that `response_mode` is set to `direct_post.jwt` in both Same Device and Cross Device flows.	`response_mode` parameter is correctly set to `direct_post.jwt` for every remote presentation transaction.
RPR-91	Response Type Validation	Test that `response_type` is set to `vp_token`.	`response_type` parameter is correctly set to `vp_token`.
RPR-92	Response URI Usage	Test that Relying Party correctly provides `response_uri` parameter to Wallet Instance.	Relying Party sends Authorization Response to correct `response_uri` endpoint.
RPR-93	Nonce Entropy	Test that nonce has sufficient entropy (32+ digits).	nonce parameter has sufficient entropy with at least 32 digits.
RPR-94	JWT Expiration	Test that JWT `exp` is set correctly.	JWT `exp` parameter is correctly set and not expired.
RPR-95	Response URI Security	Test that `response_uri` is attested by trusted third party.	`response_uri` parameter is properly attested by trusted third party.
RPR-96	Client Metadata Handling	Test that Relying Party correctly handles Wallet Instance `client_metadata`.	The `client_metadata` is correctly aligned with Trust Chain metadata.
RPR-97	Wallet Instance Attestation (WIA) Request	Test that Relying Party requests Wallet Instance Attestation (WIA) via DCQL.	Relying Party correctly requests Wallet Instance Attestation (WIA) using DCQL query.
RPR-98	Error Response Format	Test that error response uses `application/json` content type.	Error response correctly uses `application/json` content type.
RPR-99	Error Response Parameters	Test that error response includes required parameters.	Error response includes error and `error_description` parameters.
RPR-100	Wallet Instance Attestation (WIA) Presentation	Test that Relying Party correctly requests Wallet Instance Attestation (WIA) from Wallet Instance.	Relying Party correctly evaluates Wallet Instance Attestation (WIA) when requested.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-101	Presentation Array	Test that `vp_token` contains the requested signed presentations.	`vp_token` contains the requested signed presentations as required.
RPR-102	KB-JWT Inclusion	Test that Holder includes KB-JWT in SD-JWT.	Holder correctly includes KB-JWT in SD-JWT presentation.
RPR-103	KB-JWT Validation	Test that Relying Party validates KB-JWT signature.	Relying Party correctly validates KB-JWT signature using public key.
RPR-104	KB-JWT Header	Test that KB-JWT contains required header parameters.	KB-JWT contains required `typ` and `alg` header parameters.
RPR-105	KB-JWT Payload	Test that KB-JWT contains required payload parameters.	KB-JWT contains required `iat`, `aud`, `nonce`, and `sd_hash` parameters.
RPR-106	KB-JWT Audience	Test that KB-JWT `aud` matches Relying Party identifier.	KB-JWT `aud` parameter matches Relying Party unique entity identifier.
RPR-107	KB-JWT Nonce	Test that KB-JWT `nonce` matches request object `nonce`.	KB-JWT `nonce` parameter matches `nonce` from request object.
RPR-108	Authorization Error Response	Test that Relying Party correctly handles Wallet Instance Authorization Error Response on validation failure.	Relying Party correctly evaluates Authorization Error Response when validation fails.
RPR-109	Error Response Encoding	Test that Authorization Error Response is encoded correctly.	Authorization Error Response is encoded in `application/x-www-form-urlencoded` format.
RPR-110	Response Processing	Test that Response URI returns HTTP 200 on successful processing.	Response URI returns HTTP 200 with `application/json` content type.
RPR-111	Error Code Consistency	Test that error codes are consistent across different endpoints.	Error codes are consistent across all Relying Party endpoints.
RPR-112	Response Code Inclusion	Test that Relying Party includes response code in `redirect_uri`.	Relying Party includes fresh response code in `redirect_uri`.
RPR-113	Redirect URI Security	Test that `redirect_uri` is attested by trusted third party.	`redirect_uri` parameter is properly attested by trusted third party.
RPR-114	Validation Error Response	Test that Response URI returns error response on validation failure.	Response URI returns error response when validation checks fail.

continues on next page

Table 20.34 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
RPR-115	Identity Matching	Test Unique National Identifier Binding when `personal_administrative_number` or `tax_id_code` is presented.	The Relying Party uses the unique national identifier as the primary identity matching key before reconciliation, as specified in *Identity Matching and Identity Reconciliation* (page 538) (*CI_201* (page 776)).
RPR-116	Identity Matching	Test Attribute-Based Binding when a PID is presented without `personal_administrative_number`.	The Relying Party matches at least `family_name`, `given_name`, `birth_date` (or `birthdate`) and `place_of_birth` after normalisation and MUST NOT complete matching on a subset that would allow person swap, as specified in *Identity Matching and Identity Reconciliation* (page 538) (*CI_202* (page 776)).

Proximity Credential Verifier Test Matrix

This section provides the set of test cases designed for technical implementers and development teams responsible for creating and deploying Credential Verifiers solutions for proximity flows. It is also intended for assessment bodies inspecting and validating the implementations of Credential Verifiers solutions for proximity flows.

Note

Further references about official ISO-18013-5 test plans, if available, will update this section in future releases.

Test Case ID	Purpose	Description	Expected Result
PPR-001	Device Engagement	Test the initiation of device engagement using QR code.	Device engagement is successfully initiated and QR code is scanned.
PPR-002	Session Establishment	Verify session establishment with correct session keys.	Session is established securely with correct session keys.

continues on next page

Table 20.35 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
PPR-003	Communication	Test the transmission of mdoc request over BLE.	mdoc request is transmitted securely over BLE.
PPR-004	User Authentication	Validate user authentication via WSCA.	User is authenticated successfully using WSCA.
PPR-005	Attribute Consent	Check user consent for attribute release.	User consents to release requested attributes.
PPR-006	Data Retrieval	Test retrieval of mdoc Digital Credentials.	mdoc Digital Credentials are retrieved successfully.
PPR-007	Session Termination	Verify session termination after data exchange.	Session is terminated and keys are destroyed.
PPR-008	Error Handling	Test handling of invalid session keys.	Appropriate error message is displayed for invalid keys.
PPR-009	BLE Connection	Test BLE connection stability during data exchange.	BLE connection remains stable throughout the exchange.
PPR-010	Document Verification	Verify the integrity of received documents.	Documents are verified and integrity is confirmed.
PPR-011	Security	Test encryption of mdoc requests and responses.	All mdoc requests and responses are encrypted correctly.
PPR-012	User Interface	Check the user interface for attribute consent.	User interface displays attribute consent request clearly.
PPR-013	Error Handling	Test response to unsupported document types.	System returns appropriate error for unsupported document types.
PPR-014	Performance	Measure time taken for session establishment.	Session is established within acceptable time limits.
PPR-015	Compatibility	Verify compatibility with different mobile devices.	System works seamlessly across various mobile devices.
PPR-016	Data Integrity	Test integrity of data during transmission.	Data integrity is maintained during transmission.
PPR-017	Session Management	Test session management under high load.	Sessions are managed effectively under high load conditions.
PPR-018	BLE Connection	Test reconnection after BLE disconnection.	System reconnects successfully after BLE disconnection.
PPR-019	User Experience	Evaluate user experience during the proximity flow.	Users report a positive experience with the proximity flow.
PPR-020	Security	Test resistance to replay attacks.	System is resistant to replay attacks.
PPR-021	Device Engagement	Verify that Device Engagement structure is CBOR encoded.	Device Engagement structure is correctly encoded in CBOR format.

continues on next page

Table 20.35 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
PPR-022	Device Engagement	Test that ephemeral public key is of type allowed by selected cipher suite.	Ephemeral public key meets the requirements of the selected cipher suite.
PPR-023	BLE Configuration	Verify that only Central Client mode is supported.	Only Central Client mode is supported for BLE connections.
PPR-024	Capabilities	Test that `HandoverSessionEstablishmentSupport` is set to `true` when present.	`HandoverSessionEstablishmentSupport` is correctly set to `true`.
PPR-025	Capabilities	Verify that `ReaderAuthAllSupport` is set to `true` when present.	`ReaderAuthAllSupport` is correctly set to `true`.
PPR-026	mdoc Request	Test that mdoc Request messages are CBOR encoded.	mdoc Request messages are correctly encoded in CBOR format.
PPR-027	mdoc Request	Verify that mdoc request is encrypted with session key.	mdoc request is correctly encrypted with session key.
PPR-028	mdoc Request	Test that mdoc request is transmitted via BLE protocol.	mdoc request is correctly transmitted via BLE protocol.
PPR-029	mdoc Response	Verify that mdoc Response messages are CBOR encoded.	mdoc Response messages are correctly encoded in CBOR format.
PPR-030	mdoc Response	Test that mdoc response is encrypted with session key.	mdoc response is correctly encrypted with session key.
PPR-031	Key Management	Test that private ephemeral key is kept secret.	Private ephemeral key is properly secured and not exposed.
PPR-032	Key Management	Test that public ephemeral key is used in session establishment.	Public ephemeral key is correctly used for session establishment.
PPR-033	Session Key Derivation	Test that session keys are derived using key agreement protocol.	Session keys are correctly derived using the key agreement protocol.
PPR-034	Session Establishment	Test that `SessionEstablishment` message is prepared correctly.	`SessionEstablishment` message is properly prepared with required components.
PPR-035	Session Establishment	Test that `SessionEstablishment` message is signed by RP Instance.	`SessionEstablishment` message is correctly signed by Relying Party Instance.
PPR-036	Session Establishment	Test that `SessionEstablishment` message is encrypted with session keys.	`SessionEstablishment` message is properly encrypted with session keys.

continues on next page

Table 20.35 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
PPR-037	Session Establishment	Test that `SessionEstablishment` includes `EReaderKey.Pub` and attribute request.	`SessionEstablishment` message includes required `EReaderKey.Pub` and attribute request.
PPR-038	Message Transmission	Test that `SessionEstablishment` is transmitted over secure BLE connection.	`SessionEstablishment` message is transmitted over secure BLE connection.
PPR-039	Session Key Computation	Test that Relying Party correctly handles Wallet Instance session key computation.	Wallet Instance correctly computes session key.
PPR-040	Message Decryption	Test that Relying Party correctly handles Wallet Instance decrypting `SessionEstablishment` message.	Wallet Instance successfully decrypts `SessionEstablishment` message.
PPR-041	Signature Verification	Test that Relying Party correctly handles Wallet Instance verifying RP Instance signature.	Wallet Instance correctly verifies Relying Party Instance signature.
PPR-042	Attribute Request Processing	Test that Relying Party correctly handles Wallet Instance decrypting attribute request.	Wallet Instance successfully decrypts attribute request.
PPR-043	User Consent	Test that Relying Party correctly handles Wallet Instance prompting user for consent.	Wallet Instance correctly prompts user for consent to release attributes.
PPR-044	Certificate Display	Test that Relying Party correctly handles Wallet Instance displaying RP Registration Certificate.	Wallet Instance displays Relying Party Registration Certificate for transparency.
PPR-045	Credential Retrieval	Test that Relying Party correctly handles Wallet Instance retrieving requested mdoc Digital Credentials.	Wallet Instance successfully retrieves requested mdoc Digital Credentials.
PPR-046	SessionData Preparation	Test that Relying Party correctly handles Wallet Instance preparing `SessionData` message.	Wallet Instance correctly prepares `SessionData` message with Digital Credentials.
PPR-047	Authentication Data Signing	Test that Relying Party correctly handles Wallet Instance signing required authentication data.	Wallet Instance correctly signs required authentication data.
PPR-048	Message Encryption	Test that `SessionData` is encrypted with session keys.	`SessionData` message is properly encrypted with session keys.

continues on next page

Table 20.35 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
PPR-049	CBOR Encoding	Test that mdoc response is encoded in CBOR format.	mdoc response is correctly encoded in CBOR format.
PPR-050	Data Verification	Test that RP Instance decrypts `SessionData`.	Relying Party Instance successfully decrypts `SessionData`.
PPR-051	Signature Verification	Test that RP Instance verifies Wallet Instance signature.	Relying Party Instance correctly verifies Wallet Instance signature.
PPR-052	Document Validation	Test that RP Instance checks mdoc validity and Issuer signature.	Relying Party Instance correctly validates mdoc and Issuer signature.
PPR-053	BLE Disconnection	Test that GATT Client unsubscribes from characteristics.	GATT Client properly unsubscribes from characteristics.
PPR-054	BLE Disconnection	Test that GATT Client disconnects from GATT server.	GATT Client properly disconnects from GATT server.
PPR-055	Request Structure Compliance	Test that mdoc Request is compliant with required structure.	mdoc Request complies with required structure and includes necessary components.
PPR-056	Response Structure Compliance	Test that mdoc Response is compliant with required structure.	mdoc Response complies with required structure and includes necessary components.
PPR-057	Document Structure Compliance	Test that documents are compliant with required structure.	Documents comply with required structure and include necessary components.
PPR-058	Document Type Validation	Test that mDL document type is correctly set.	mDL document type is correctly set to `org.iso.18013.5.1.mDL`.
PPR-059	DeviceSigned Structure	Test that deviceSigned structure is compliant.	deviceSigned structure complies with required format and includes necessary components.
PPR-060	Device Authentication	Test that `deviceAuth` includes `deviceSignature`.	`deviceAuth` structure includes required `deviceSignature` for authentication.
PPR-061	Wallet Instance Attestation (WIA) Inclusion	Test that Relying Party correctly handles Wallet Instance including Wallet Instance Attestation (WIA) when requested.	Wallet Instance includes Wallet Instance Attestation (WIA) when requested by Relying Party.
PPR-062	AAL Claim Inclusion	Test that Relying Party correctly handles Wallet Instance including `aal` claim in Wallet Instance Attestation (WIA).	Wallet Instance includes `aal` claim as disclosure in Wallet Instance Attestation (WIA).

continues on next page

Table 20.35 – continued from previous page

Test Case ID	Purpose	Description	Expected Result
PPR-063	User Consent Bypass	Test that Relying Party correctly handles Wallet Instance not requesting user consent for Wallet Instance Attestation (WIA).	Wallet Instance does not request user consent for technical Wallet Instance Attestation (WIA) attributes.
PPR-064	Session Termination Conditions	Test that session is terminated under specified conditions.	Session is properly terminated when specified conditions occur.
PPR-065	Session Termination Initiation	Test that session termination is initiated correctly.	Session termination is properly initiated when no further requests are sent.
PPR-066	Key Destruction	Test that session keys are destroyed on termination.	Session keys and ephemeral key material are properly destroyed.
PPR-067	Channel Closure	Test that communication channel is closed on termination.	Communication channel used for data retrieval is properly closed.

20.5 Wallet Provider PDND OpenAPI Specification

Below is the complete OpenAPI Specification for the Wallet Provider PDND e-services:

```yaml
openapi: 3.0.1
info:
  title: IT Wallet API - Wallet Provider web services
  version: 0.2.0
  description: IT Wallet Provider e-Service exposed via PDND.
  termsOfService: "https://authentic-source.example.it/tos/"
  contact:
    name: IT-Wallet <wallet_provider>
    url: https://github.com/italia/eid-wallet-it-docs
  x-api-id: WPITW-01
  x-summary: IT Wallet Wallet Provider API.
servers:
  - url: https://test.wallet-provider.example.it/v0.2.0
    description: Wallet Provider API test server
  - url: https://wallet-provider.example.it/v0.2.0
    description: Wallet Provider API production server
paths:
  /status:
    get:
      tags:
        - status
```

```yaml
      summary: Get Wallet Provider API status.
      description: Health-check endpoint that returns the operational status of the Wallet Provider API.
      operationId: walletProviderStatus
      responses:
        "200":
          description: Service available
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Cache-Control:
              $ref: "#/components/headers/CacheControlHeader"
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "429":
          description: Too Many Requests
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
          # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X-RateLimit-Reset are not required because redundant along with Retry-After.
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"
        "503":
          description: Service Unavailable
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"

  /wallet-instances:
    patch:
      tags:
        - instance
      summary: Notification of User's death.
      description: >-
```

```
67          This service is used to notify the Wallet Provider of the need to revoke the Wallet Ins
    tance and delete the User's account due to the User's death.
68        operationId: notifyUserDeath
69        parameters:
70          - name: Agid-JWT-Signature
71            in: header
72            description: >-
73              JWT containing the signature of the message headers whose integrity
74              needs to be guaranteed, to comply with the INTEGRITY_REST_02
75              security pattern (see <a target="blank"
76              href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pd
    nd.html">e-Service PDND</a>). <br/><br/>
77
78              <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZ
    CI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi0
    2MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5Mz
    MiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxN
    zMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIs
    InNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1Yzl
    mMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.t
    G5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA">EXAMPLE
79              ON JWT.IO</a>
80            required: true
81            schema:
82              type: string
83              format: JWT
84              example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NT
    QzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4
    MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXh
    hbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZj
    diMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2P
    TcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNv
    bnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5S
    m2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA
85          - name: Digest
86            in: header
87            description: >-
88              Digest of the message payload, to comply with the INTEGRITY_REST_02
89              security pattern. According to <a target="blank" href="https://www.rfc-editor.org/r
    fc/rfc3230.html#section-4.2">RFC
90              3230 §4.2</a>, the format MUST be the following: digest-algorithm=encoded
91              digest output.
92            required: true
93            schema:
94              type: string
95              example: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f41
96          - name: Agid-JWT-TrackingEvidence
```

```
97            in: header
98            description: >-
99              TrackingEvidence JWT for AUDIT_REST_02. Optional. It MAY be sent
100             together with POP_DPoP (`Authorization: DPoP` and the `DPoP` proof)
101             when the Provider requires tracked data from the Consumer, such as
102             caller LoA, userID, IP address, or other data agreed with the
103             Provider. The DPoP proof remains the proof of possession. When the
104             Provider mandates AUDIT_REST_02, this header MUST be present.
105             For POP_TPoP the same header is required by `TrackingEvidenceAuth`
106             together with `BearerAuth`; send a single header value
107             (see <a target="blank"
108             href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pd
    nd.html">e-Service PDND</a>). <br/><br/>
109             <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZ
    CI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi0
    2MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdC
    IsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4Z
    jktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUt
    ZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJ
    zdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInnc
    dhjXhUjLS1V76cGBxkutDNvxRNA">EXAMPLE
110             ON JWT.IO</a>
111           required: false
112           schema:
113             type: string
114             format: JWT
115             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NT
    QzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJo
    dHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJ
    pYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6Nj
    UyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEI
    joiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0
    b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA
116       security:
117         - DPoPVoucherAuth: []
118           DPoPProof: []
119         - BearerAuth: []
120           TrackingEvidenceAuth: []
121       requestBody:
122         required: true
123         content:
124           application/merge-patch+json:
125             schema:
126               $ref: "#/components/schemas/NotifyUserDeath"
127       responses:
128         "207":
129           description: Multi-Status
```

```
130           headers:
131             Agid-JWT-Signature:
132               description: JWT containing the signature of the message headers whose integrity
    needs to be guaranteed, to comply with the INTEGRITY_REST_02 security pattern (see <a target="b
    lank" href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pdnd.htm
    l">e-Service PDND</a>). <a target="blank" href="https://jwt.io/#debugger-io?token=ew0KICAiYWxnI
    jogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYjBkMi03OWUzZjBjNGE4ZWYiLA0KICAidHlwIjog
    IkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3OC1pOWowLWtsbW5vcHFyc3R3eCIsDQogICJzdWIiOiA
    iMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdHd4IiwNCiAgImF1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW
    1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6IDE3MzMzOTc4NDAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZ
    XhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6ICI4ZTEyZjRiNy05YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAi
    c2lnbmVkX2hlYWRlcnMiOiBbDQogICAgew0KICAgICAgImRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTg
    zMDYwMGFkOWJkY2E5OTM1OTNmODc2MjA5YTAwNGI1OTliNTgzMDk1YjBhNjEiDQogICAgfSwNCiAgICB7DQogICAgICAiY2
    9udGVudC10eXBlIjogImFwcGxpY2F0aW9uL2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPv
    CX6HfktkxG7L5mk6a8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg">EXAMPLE
133               required: true
134               schema:
135                 type: string
136             Digest:
137               description: Digest of the message payload, to comply with the INTEGRITY_REST_02
    security pattern. According to RFC 3230 Section 4.2 <a target="blank" href="https://www.rfc-edi
    tor.org/rfc/rfc3230.html#section-4.2">RFC 3230 §4.2</a>, the format MUST be the following diges
    t-algorithm=encoded digest output.
138               required: true
139               schema:
140                 type: string
141                 example: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b583095b0a
    61
142             Cache-Control:
143               $ref: "#/components/headers/CacheControlHeader"
144             X-RateLimit-Limit:
145               $ref: "#/components/headers/RateLimitLimitHeader"
146             X-RateLimit-Remaining:
147               $ref: "#/components/headers/RateLimitRemainingHeader"
148             X-RateLimit-Reset:
149               $ref: "#/components/headers/RateLimitResetHeader"
150           content:
151             application/json:
152               schema:
153                 $ref: "#/components/schemas/EServiceResponse"
154         "400":
155           description: Bad Request
156           content:
157             application/problem+json:
158               schema:
159                 $ref: "#/components/schemas/ProblemDetails"
160           headers:
```

```yaml
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "401":
          description: Unauthorized
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
            WWW-Authenticate:
              $ref: "#/components/headers/WWWAuthenticateHeader"
        "404":
          description: Wallet Instance identifiers not found
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "429":
          description: Too Many Requests
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X-Rate
Limit-Reset are not required because redundant along with Retry-After.
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"
        "500":
          description: Internal Server Error.
```

```yaml
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"
        "503":
          description: Service Unavailable
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Retry-After:
              $ref: "#/components/headers/RetryAfterHeader"

tags:
  - name: status
    description: Endpoint di health check dell'API.
  - name: instance
    description: Retrieve information about the user's death.

components:
  securitySchemes:
    DPoPVoucherAuth:
      type: http
      scheme: DPoP
      description: >-
        PDND Voucher issued with DPoP binding (POP_DPoP, RFC 9449). The
        `Authorization` header MUST be `DPoP <voucher>` (`token_type` is `DPoP`).
        This scheme MUST be used together with `DPoPProof`. It MUST NOT be
        modelled as HTTP Bearer: a conformant Provider rejects
        `Authorization: Bearer` for a DPoP-bound Voucher.
    DPoPProof:
      type: apiKey
      in: header
      name: DPoP
      description: >-
        DPoP proof JWT (RFC 9449). REQUIRED together with `DPoPVoucherAuth`
        (POP_DPoP). MUST NOT be sent with `BearerAuth` (POP_TPoP), where proof
        of possession is `Agid-JWT-TrackingEvidence`.
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: >-
```

```
254         PDND Voucher issued as Bearer (POP_TPoP). The `Authorization` header
255         MUST be `Bearer <voucher>` (`token_type` is `Bearer`). This scheme MUST
256         be used together with `TrackingEvidenceAuth` as proof of possession.
257         This is REST_JWS_2021_POP via TrackingEvidence, not REST_JWS_2021_Bearer.
258     TrackingEvidenceAuth:
259       type: apiKey
260       in: header
261       name: Agid-JWT-TrackingEvidence
262       description: >-
263         TrackingEvidence JWT. REQUIRED together with `BearerAuth` (POP_TPoP) as
264         proof of possession of the PDND Voucher. The same header is also an
265         optional operation parameter, so it MAY be sent with the POP_DPoP pair
266         (`DPoPVoucherAuth` + `DPoPProof`) when the Provider requires
267         AUDIT_REST_02. The DPoP proof remains the proof of possession.
268         See e-Service PDND in the Technical Specifications.
269
270   headers:
271     CacheControlHeader:
272       schema:
273         type: string
274         enum:
275           - no-store
276       description: no-store
277     RateLimitLimitHeader:
278       schema:
279         type: integer
280         format: int32
281         minimum: 0
282       description: Maximum number of requests within the time window.
283     RateLimitRemainingHeader:
284       schema:
285         type: integer
286         format: int32
287         minimum: 0
288       description: Remaining requests within the time window.
289     RateLimitResetHeader:
290       schema:
291         type: integer
292         format: int32
293         minimum: 0
294       description: UTC epoch in seconds, corresponding to when the window for the current rate
    limit will reset.
295     RetryAfterHeader:
296       schema:
297         type: integer
298         format: int32
299         minimum: 0
```

```
300       description: Seconds to wait before receiving another response.
301     WWWAuthenticateHeader:
302       schema:
303         type: string
304         example: >-
305           Bearer error="invalid_token", error_description="The access token expired"
306       description: The request cannot be fulfilled because the Voucher is expired, revoked or o
    therwise malformed. See <a target="blank" href="https://datatracker.ietf.org/doc/html/rfc6750.h
    tml#section-3">RFC6750</a> and <a target="blank" href="https://datatracker.ietf.org/doc/html/rf
    c9449.html#section-7.1-11">RFC9449</a> for details.
307
308   schemas:
309     EServiceResponse:
310       properties:
311         result:
312           type: object
313           properties:
314             revoked:
315               type: array
316               items:
317                 type: string
318               description: List of Wallet Instances successfully rekoved.
319             not_found:
320               type: array
321               items:
322                 type: string
323               description: List of Wallet Instances not revoked as their identifier were not fo
    und at the Wallet Provider.
324             already_revoked:
325               type: array
326               items:
327                 type: string
328               description: List of Wallet Instances that were already revoked by the Wallet Pro
    vider.
329           required: [revoked, not_found, already_revoked]
330           description: >-
331             JSON object specifying which Wallet Instance was successfully revoked, which was al
    ready revoked and which was not found.
332         result_description:
333           description: Response Description.
334           type: string
335           example: The service partially process the revocation of the requested Wallet Instanc
    es as some identifiers were not found.
336           required: [result, result_description]
337     ProblemDetails:
338       type: object
339       description: RFC7807-compliant problem details object for error responses.
```

```yaml
      properties:
        type:
          type: string
          format: uri
          description: An absolute URI that identifies the problem type.
        title:
          type: string
          description: A short, human-readable summary of the problem type.
        status:
          type: integer
          format: int32
          description: The HTTP status code generated by the origin server for this occurrence
of the problem.
        detail:
          type: string
          description: A human-readable explanation specific to this occurrence of the problem.
        instance:
          type: string
          format: uri
          description: An absolute URI that identifies the specific occurrence of the problem.
      required: [title, status, detail]
    NotifyUserDeath:
      required:
        - wallet_instance_ids
      type: object
      properties:
        wallet_instance_ids:
          type: array
          items:
            type: string
          description: >-
            Identifiers of the Wallet Instances whose PID was revoked due to User death.
```

INDEX

A

Access CA, **639**
Access Certificate, **639**
Access Certificate Authority, **639**
Accreditation Process, **639**
AS Registry, **640**
Attestation Provider, **640**
Attributes, **639**
Authentic Source, **640**
Authentic Source Registry, **640**
Authentication, **640**
Authentication Button, **650**

B

Brand Identity, **650**
Brand Manual, **650**

C

Call To Action, **650**
Catalog, **650**
Certificate Signing Request, **640**
Certification Process, **640**
Conformity Assessment Body, **640**
Credential, **641**
Credential Issuer, **640**
Credential of Public Interest, **641**
Credential Verifier, **648**
Critical Assets, **640**
Cryptographic Hardware Key Tag, **640**
Cryptographic Hardware Keys, **640**

D

Dataset_id, **640**
Detailed View, **650**
Device Integrity Service, **641**
Digital Credential, **641**
Digital Credential Catalog, **641**
Discovery Page, **650**

E

eIDAS Trusted List, **641**
Electronic Attestation of Attributes, **641**
Electronic Attestation of Attributes issued by or on behalf of a public sector body, **641**
Electronic Attestation of Attributes Provider, **641**
Electronic Attestation of Person Identification Data, **643**
Electronic Attestation of Public Interest, **641**
Electronic Attestation Provider, **641**
Engagement Button, **650**

F

Federation Authority, **642**
Federation Registry, **642**

H

Holder, **642**
Holder Key Binding, **642**

I

Identity and Access Management, **642**
Identity Matching, **642**
Identity Reconciliation, **642**
Interaction Model, **650**

Interface, **650**
Intermediary, **642**
Intermediate Entity, **642**
Issuer, **640**
IT-Wallet ID, **643**
IT-Wallet Intermediary, **643**
IT-Wallet System, **643**
IT-Wallet System Register, **643**

K

KA, **643**
Key Attestation, **643**
Key Attestation APIs (OEM), **644**
Keystore, **644**

L

Level of Assurance, **644**
Local Internal hardware-backed Keystore, **644**

M

Member State Trusted List Provider, **644**
Metadata, **644**
MS TLP, **644**

N

National Accreditation Bodies, **644**
National Identity Provider, **645**
National Trust Anchor, **645**
Notification Process, **645**

O

Object_id, **645**
OpenID Federation Intermediate, **642**
Organizational Entity, **645**

P

Person Identification Data, **645**
PID, **645**
PID Provider, **645**
Policy Language, **645**
Preview View, **650**
Primary Actors, **645**
Provider of Person Identification Data, **645**
Provider of Registration Certificates, **646**
Pseudonym, **646**
Public Electronic Attestation of Attributes, **641**

Q

Qualified Electronic Attestation of Attributes, **646**
Qualified Electronic Attestation of Attributes Provider, **646**
Qualified Electronic Signature Provider, **646**

R

Reg. Cert. Provider, **646**
Registrar, **646**
Registration Authority, **646**
Registration Certificate, **646**
Registration Certificate Provider, **646**
Registration Process, **646**
Relying Party, **646**
Relying Party Instance, **646**
Relying Party Intermediary, **646**
Relying Party Provider Backend, **647**
Relying Party Solution, **647**
RFC
- RFC 1950, 411, 417, 805
- RFC 1951, 411, 417, 805
- RFC 2511, 601
- RFC 3230, 704, 706
- RFC 3280, 266
- RFC 3339, 379
- RFC 3986, 415
- RFC 4122, 569, 578
- RFC 4648, 759
- RFC 5280, 194, 359, 413, 551, 558, 655
- RFC 5646, 338, 339
- RFC 5869, 616
- RFC 6749, 422, 423, 432, 496, 565, 567, 576–578, 687–689, 749, 785, 786
- RFC 67491, 577

RFC 6749#section-4.1.2, 574
RFC 6749#section-4.1.2.1, 497, 498, 510, 571, 574, 575, 596
RFC 6749#section-5.2, 571, 579, 580, 690, 691, 696
RFC 6750, 587, 592, 788, 789
RFC 6750#section-3, 696, 707
RFC 6750#section-3.1, 697
RFC 6949, 790
RFC 7231, 546, 745
RFC 7515, 194, 336, 359, 361, 413, 551, 558, 577, 578, 580–585, 688–690, 704, 705
RFC 7516, 615, 616, 618
RFC 7516#section-4.1.1, 303, 498, 548, 551, 554, 557, 566, 569
RFC 7517, 118, 413, 498, 558, 578, 585, 692, 693, 695, 696
RFC 7518, 615–618
RFC 7519, 118, 359, 360, 413, 417, 428, 498, 548, 552, 555, 558, 566–570, 574, 578, 580–583, 585, 688, 690, 704–706
RFC 7521, 574, 687, 688
RFC 7523, 687, 688
RFC 7591#section-2, 355
RFC 7591#section-3.2.1, 355
RFC 7636, 336, 422, 427, 433, 729, 748, 785
RFC 7636#section-4.2, 567
RFC 7636#section-4.3, 567
RFC 7638, 581, 583, 690
RFC 7638#section_3, 413, 548, 551, 555, 557, 566, 581, 582
RFC 7800, 548, 552, 555
RFC 7807, 546, 745
RFC 8259, 570, 783
RFC 8414, 336, 338, 565
RFC 8414#section-2, 336, 337
RFC 8725, 566, 577
RFC 8949, 366, 388, 795
RFC 8963, 690
RFC 9027, 432
RFC 9052, 367, 797
RFC 9053, 367
RFC 9068, 581–583, 689, 690
RFC 9101, 337, 422, 427, 428, 430, 498, 506, 566, 778, 779
RFC 9110, 416, 578, 589, 590, 807
RFC 9126, 422, 427, 428, 431, 548, 552, 555, 558, 566, 567, 569, 570, 574, 748, 778–780, 783, 784
RFC 9126#as_metadata, 336
RFC 9126#section-2.3, 570, 571
RFC 9207, 574, 749
RFC 9360, 367
RFC 9396, 422, 427, 568, 748
RFC 9449, 337, 432, 433, 436, 440, 447, 448, 576, 580–584, 588, 677, 680, 687, 689, 690, 696, 699, 700, 703, 786, 788
RFC 9449#section-4, 749, 750
RFC 9449#section-5, 691
RFC 9449#section-7.1, 707
RFC 9457, 707
RFC 9864, 615

S

Selection Page, **651**
Selective Disclosure, **647**
Self-Sovereign Identity, **647**
Service Model, **651**
Signal (*Signal Hub*), **647**
Signal Hub, **647**
Supervision Process, **647**

T

Technical Solutions, **647**
Technical Specifications, **648**
Touchpoint, **651**
Trust, **648**
Trust Attestation, **648**
Trust Evaluation, **648**
Trust Framework, **648**
Trust Layer, **648**
Trust Mark, **651**
Trust Model, **648**

Trust Relationship, **648**
Trusted List, **648**

U

User, **648**
User Attribute, **639**
User Claim, **639**
User Experience, **651**

V

Verifier, **648**
Verifier App, **646**
Visual Identity, **651**

W

Wallet Instance, **648**
Wallet Instance Attestation, **649**
Wallet Provider, **649**
Wallet Provider Backend, **649**
Wallet Secure Cryptographic Application, **649**
Wallet Secure Cryptographic Device, **649**
Wallet Solution, **649**
Wallet Unit, **649**
Wallet Unit Attestation, **649**
Wallet-Relying Party, **646**
WIA, **649**
WUA, **649**

Specifiche Tecniche IT-Wallet

Versione 1.4.7

Data di rilascio: 22/09/2026

Indice dei Contenuti

1 Introduzione **3**
- 1.1 Ambito . . . 5
- 1.2 Linguaggio Normativo e Convenzioni . . . 5
- 1.3 Come leggere le Specifiche . . . 6
  - 1.3.1 Panoramica della Struttura delle Specifiche . . . 6
  - 1.3.2 Percorsi di lettura per ruolo . . . 7

2 Brand Identity **15**
- 2.1 Naming . . . 16
- 2.2 Identità Visiva . . . 16
  - 2.2.1 Logo . . . 16
  - 2.2.2 Trust Mark . . . 18

3 Panoramica dell'Architettura **21**

4 Sistema di Onboarding **25**
- 4.1 Architettura del Sistema di Onboarding . . . 25
- 4.2 Processo di Registrazione della Fonte Autentica . . . 27
- 4.3 Processo di Onboarding della Federazione . . . 28
  - 4.3.1 Onboarding degli Intermediari di Relying Party . . . 29
- 4.4 Gestione del Ciclo di Vita delle Entità . . . 30
  - 4.4.1 Gestione delle Operazioni Continuative . . . 30
  - 4.4.2 Processi di Uscita e Rimozione dalla Federazione . . . 31
  - 4.4.3 Requisiti di Coordinamento del Ciclo di Vita . . . 32
- 4.5 Onboarding Journey Maps . . . 32
  - 4.5.1 Panoramica dell'Ecosistema . . . 32
  - 4.5.2 Journey dell'Operatore della Fonte Autentica . . . 34
  - 4.5.3 Journey dell'Operatore del Credential Issuer . . . 35

4.5.4 Journey dell'Operatore del Fornitore di Wallet . . . 36
4.5.5 Journey dell'Operatore della Relying Party . . . 37
4.6 Journey dell'Esperienza Utente . . . 38

5 Design dell'Esperienza Utente 39
5.1 Principi di Design . . . 39
5.2 Panoramica delle Funzionalità . . . 40
5.3 Attivazione dell'Istanza del Wallet . . . 41
5.3.1 Focus sugli Attestati di Identificazione Personale . . . 42
5.4 Ottenimento degli Attestati Elettronici di Attributi . . . 46
5.4.1 Ottenimento dal Catalogo dell'Istanza del Wallet . . . 47
5.4.2 Ottenimento dal Touchpoint della Fonte Autentica . . . 49
5.4.3 Focus sugli Attestati Elettronici di Attributi . . . 61
5.5 Presentazione degli Attestati Elettronici . . . 67
5.5.1 Presentazione in prossimità . . . 69
5.5.2 Presentazione da remoto . . . 70
5.6 Gestione degli Attestati Elettronici . . . 100
5.6.1 Stato degli Attestati Elettronici . . . 100
5.6.2 Registrazione delle Transazioni . . . 106
5.6.3 Archiviazione e ripristino degli Attestati Elettronici di Attributi . . . 107
5.7 Disattivazione dell'Istanza del Wallet . . . 107
5.8 Gestione degli errori . . . 109
5.8.1 Errori di Attivazione dell'Istanza del Wallet . . . 110
5.8.2 Errori di ottenimento degli Attestati Elettronici di Attributi . . . 111
5.8.3 Errori di presentazione degli Attestati Elettronici . . . 111
5.8.4 Errori di gestione degli Attestati Elettronici . . . 112
5.8.5 Errori di disattivazione dell'Istanza del Wallet . . . 112
5.9 Assistenza Utente . . . 112
5.10 Feedback Utente . . . 114

6 L'Infrastruttura di Trust 117
6.1 Ruoli di Federazione . . . 118
6.2 Integrazione dell'Infrastruttura di Trust e del Registro . . . 120
6.3 Proprietà Generali . . . 120
6.4 Requisiti dell'Infrastruttura di Trust . . . 121
6.5 Endpoint API di Federazione . . . 123
6.6 Configurazione della Federazione . . . 125
6.7 Entity Configuration . . . 126
6.7.1 Parametri Comuni delle Entity Configuration . . . 127
6.7.2 Entity Configuration Trust Anchor . . . 128
6.7.3 Trust Mark degli Intermediari di Relying Party . . . 128
6.7.4 Entity Configuration Foglie e Intermediari . . . 130

6.7.5 Tipi di Metadati . . . 130
6.8 Metadati di federation_entity Foglie . . . 131
6.9 I Subordinate Statement . . . 132
6.9.1 Subordinate Statement . . . 134
6.10 Discovery della Federazione . . . 135
6.11 Trust Chain . . . 136
6.11.1 Meccanismi di Attestazione di Trust Offline . . . 139
6.11.2 Rinnovo Rapido della Trust Chain . . . 140
6.12 Meccanismo di Trust Evaluation . . . 140
6.12.1 Stabilire Trust con i Credential Issuer . . . 142
6.12.2 Stabilire Trust con il Relying Party . . . 144
6.12.3 Valutare Trust con i Wallet . . . 147
6.13 Non-ripudio delle Attestazioni di Lunga Durata . . . 147
6.14 X.509 PKI . . . 147
6.14.1 Trust Anchor di Federazione e CA X.509 . . . 149
6.14.2 Emissione di Certificati X.509 . . . 149
6.14.3 Revoca di Certificati X.509 . . . 155
6.14.4 Federation Subordinate Events Endpoint . . . 156
6.15 Osservazioni sulla Privacy . . . 157
6.16 Considerazioni sulla Decentralizzazione . . . 158

7 Infrastruttura del Registro **159**
7.1 Panoramica dell'Architettura di Registro . . . 159
7.2 Endpoint di Discovery del Registro . . . 160
7.2.1 Parametri dell'Endpoint di Discovery del Registro . . . 161
7.3 Registro dei Claims . . . 162
7.3.1 Utilizzo del Registro dei Claims . . . 163
7.3.2 Struttura del Registro dei Claims . . . 164
7.4 Registro delle Fonti Autentiche . . . 173
7.4.1 Utilizzo del Registro delle Fonti Autentiche . . . 174
7.4.2 Coordinamento FA Pubblica vs FA Privata . . . 175
7.4.3 Struttura del Registro FA . . . 175
7.4.4 Coordinamento FA-EI . . . 189
7.5 Registro della Federazione . . . 189
7.5.1 Ruolo di Integrazione del Registro . . . 189
7.5.2 Accesso al Registro della Federazione . . . 189
7.6 Catalogo degli Attestati Elettronici . . . 190
7.6.1 Gerarchia delle Credenziali Digitali . . . 194
7.6.2 Struttura del Catalogo degli Attestati Elettronici . . . 206
7.6.3 Decentralizzazione delle Informazioni di Visualizzazione e dei Claims . . . 215
7.7 Tassonomia . . . 215

7.8 Registro degli Schema . . . 232
7.9 Integrazione del Registro e Riferimenti Incrociati . . . 234
7.10 Percorsi di Utilizzo dell'Infrastruttura del Registro . . . 235
7.10.1 Navigazione del Catalogo . . . 235
7.10.2 Emissione di Credenziali . . . 235
7.10.3 Presentazione e Verifica delle Credenziali . . . 236

8 Onboarding delle Entità 239
8.1 Panoramica . . . 239
8.1.1 Architettura del Sistema di Onboarding delle Entità . . . 239
8.1.2 Tipologie di Entità e Percorsi di Onboarding . . . 240
8.1.3 Registrazione Amministrativa e Tecnica . . . 241
8.2 Processo di Registrazione delle Fonti Autentiche . . . 242
8.2.1 Requisiti di Registrazione delle Fonti Autentiche . . . 242
8.2.2 Requisiti sulle informazioni di registrazione delle Fonti Autentiche . . . 243
8.2.3 Procedura di registrazione delle Fonti Autentiche . . . 244
8.3 Processo di Autorizzazione dalla Fonte Autentica all'Emittente di Credenziali . . . 255
8.4 Processo di Onboarding delle Entità Federate . . . 256
8.4.1 Modello Gerarchico dell'Autorità di Federazione . . . 256
8.4.2 Prerequisiti di Onboarding alla Federazione . . . 257
8.4.3 Requisiti di Sicurezza per la Gestione delle Chiavi . . . 258
8.4.4 Procedura di Onboarding alla Federazione . . . 258
8.4.5 Trust Mark della Federazione IT-Wallet . . . 266

9 Operazioni di Gestione dei Certificati X.509 271
9.1 Architettura PKI della Federazione . . . 271
9.2 Struttura e Analisi della Catena di Certificati X.509 . . . 272
9.2.1 Visualizzazione della Catena di Certificati X.509 . . . 272
9.2.2 Validazione della Catena di Certificati X.509 . . . 273
9.3 Gestione della Revoca dei Certificati X.509 . . . 274
9.3.1 Distribuzione e Accesso CRL . . . 274
9.3.2 Verifica della Revoca dei Certificati X.509 . . . 275
9.4 Migliori Pratiche per la Gestione dei Certificati X.509 . . . 276
9.4.1 Integrazione della Validazione dei Certificati X.509 . . . 276
9.4.2 Diagnostica e Risoluzione dei Problemi . . . 276
9.4.3 Coordinamento del Ciclo di Vita dei Certificati X.509 . . . 278
9.5 Procedure di Rotazione delle Chiavi . . . 278
9.5.1 Rotazione delle Chiavi Superiori . . . 278
9.5.2 Rotazione delle Chiavi Subordinate . . . 280
9.6 Aggiornamenti della Configurazione Tecnica . . . 281
9.7 Procedure Tecniche di Uscita dalla Federazione . . . 283
9.7.1 Uscita Volontaria - Disattivazione Tecnica . . . 283

9.7.2 Rimozione da Parte dell'Organo di Supervisione - Implementazione Tecnica 284

10 Entità **287**

10.1 Soluzione Wallet . . . 287
10.1.1 Requisiti della Soluzione Wallet . . . 289
10.1.2 Componenti della Soluzione Wallet . . . 294
10.1.3 Modelli di Interazione della Soluzione Wallet . . . 298
10.1.4 Istanza del Wallet . . . 298
10.1.5 Backup e Ripristino . . . 320
10.1.6 Entity Configuration del Fornitore di Wallet . . . 326
10.1.7 Metadati della Soluzione Wallet . . . 329
10.2 Soluzione del Fornitore di Attestati Elettronici . . . 330
10.2.1 Requisiti del Fornitore di Attestati Elettronici . . . 331
10.2.2 Dettagli dei Componenti . . . 332
10.2.3 Modelli di Interazione . . . 334
10.2.4 Entity Configuration del Fornitore di Attestati Elettronici . . . 334
10.2.5 Metadata del Fornitore di Attestati Elettronici . . . 356
10.3 Soluzione di Relying Party . . . 361
10.3.1 App di Verifica . . . 363
10.3.2 App di Verifica Mobile . . . 364
10.3.3 App di Verifica Web . . . 371
10.3.4 Entity Configuration Relying Party . . . 372
10.3.5 Metadati della Relying Party . . . 375
10.4 Fonti Autentiche . . . 376

11 Gestione degli Attestati Elettronici **379**

11.1 Modello di Dati degli Attestati Elettronici . . . 379
11.1.1 Attributi di Metadati Format-Agnostic dell'Attestato Elettronico . . . 380
11.1.2 Formato Attestato Elettronico SD-JWT-VC . . . 380
11.1.3 Formato Attestato Elettronico mdoc-CBOR . . . 389
11.1.4 Mappatura dei Parametri degli Attestati Elettronici tra i vari Formati . . . 402
11.2 Modello di Dati del PID . . . 404
11.2.1 Modello Dati PID in formato SD-JWT-VC . . . 405
11.2.2 Modello Dati PID in formato mdoc-CBOR . . . 411
11.3 Modello di Dati dell'IT-Wallet ID . . . 416
11.3.1 Modello di Dati dell'IT-Wallet ID in formato SD-JWT VC . . . 417
11.4 Ciclo di Vita degli Attestati Elettronici . . . 422
11.4.1 Transizioni degli Attestati Elettronici . . . 424
11.4.2 Ciclo di vita degli Attestati Elettronici ottenuti in batch . . . 426
11.4.3 Gestione del Ciclo di Vita degli Attestati Elettronici . . . 426
11.4.4 Meccanismi di Verifica della Validità . . . 434

12 Flussi relativi agli Attestati Elettronici **443**
12.1 Emissione di Attestati Elettronici . . . 443
12.1.1 Flussi ad Alto Livello per l'Emissione di Attestati Elettronici . . . 443
12.1.2 Flussi Dettagliati per l'Emissione di Attestati Elettronici . . . 447
12.1.3 Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID . . . 482
12.2 Presentazione dell'Attestato Elettronico . . . 506
12.2.1 Flusso Remoto . . . 509
12.2.2 Flusso di Prossimità . . . 538
12.2.3 Identity Matching e Identity Reconciliation . . . 567
12.3 Flusso di Recupero dei Wallet Metadata . . . 570

13 Endpoints **571**
13.1 Endpoint del Fornitore di Wallet . . . 571
13.1.1 Endpoint di Federazione . . . 572
13.1.2 Endpoint Nonce della Soluzione Wallet . . . 572
13.1.3 Endpoint di Gestione dell'Istanza di Wallet . . . 572
13.1.4 Endpoint di Emissione della Wallet Instance Attestation . . . 576
13.1.5 Endpoint di Emissione della Key Attestation . . . 583
13.1.6 Catalogo e-Service PDND del Fornitore di Wallet . . . 591
13.2 Endpoint del Credential Issuer . . . 592
13.2.1 Endpoint Metadata . . . 592
13.2.2 Endpoint di Rilascio degli Attestati Elettronici . . . 595
13.2.3 Catalogo degli e-Service PDND del Credential Issuer . . . 624
13.3 Endpoint della Relying Party . . . 624
13.3.1 Endpoint per Flussi Remoti della Relying Party . . . 624
13.3.2 Endpoint del Backend del Provider di Relying Party . . . 628
13.4 Endpoint delle Fonti Autentiche . . . 634
13.4.1 Catalogo degli e-Service PDND delle Fonti Autentiche . . . 634
13.5 Endpoint di PDND Signal Hub . . . 639
13.5.1 Prerequisiti . . . 640
13.5.2 e-Service di Signal Hub . . . 640
13.5.3 Elaborazione dei Segnali . . . 644

14 Algoritmi Crittografici **647**

15 Considerazioni di Sicurezza e Privacy **653**
15.1 Requisiti di Sicurezza . . . 654
15.1.1 SR-CF-10 e SR-E-10 . . . 654
15.1.2 SR-CF-20 . . . 655
15.1.3 SR-CF-21 . . . 655
15.1.4 SR-E-20 . . . 655

15.1.5 SR-E-30 . . . 656
15.1.6 SR-E-40 . . . 656
15.1.7 SR-I-10 . . . 656
15.1.8 SR-I-20 . . . 657
15.1.9 SR-I-30 . . . 657
15.1.10 SR-I-40 . . . 657
15.1.11 SR-I-50 . . . 658
15.1.12 SR-P-20 . . . 658
15.1.13 SR-P-30 . . . 659
15.1.14 SR-P-40 . . . 659
15.1.15 SR-P-41 . . . 660
15.1.16 SR-P-50 . . . 660
15.1.17 SR-V-10 . . . 661
15.1.18 SR-V-20 . . . 662
15.1.19 SR-W-20 . . . 662
15.1.20 SR-W-30 . . . 662
15.2 Requisiti di Privacy . . . 663
15.2.1 PR-CF-30 . . . 663
15.2.2 PR-CF-40 . . . 663
15.2.3 PR-E-60 . . . 663
15.2.4 PR-E-70 . . . 664
15.2.5 PR-W-40 . . . 664
15.2.6 PR-W-60 . . . 665
15.2.7 PR-W-70 . . . 665
15.3 Requisiti di Sicurezza e Privacy . . . 665
15.3.1 SPR-E-50 . . . 665
15.3.2 SPR-P-10 . . . 666
15.3.3 SPR-P-60 . . . 667
15.3.4 SPR-P-70 . . . 667
15.3.5 SPR-P-80 . . . 669
15.3.6 SPR-W-50 . . . 669

16 Politiche Generali di Conservazione dei Log **671**
16.1 Requisiti Generali di Logging ISO/IEC 27001 . . . 672
16.1.1 Politica di Conservazione dei Log del Fornitore di Wallet . . . 673
16.1.2 Politica di Conservazione dei Log del Fornitore di Credenziali . . . 673
16.1.3 Politica di Conservazione dei Log delle Fonti Autentiche . . . 673
16.1.4 Politica di Conservazione dei Log della Relying Party . . . 673
16.1.5 Funzionalità di Logging delle Istanze del Wallet . . . 674

17 Termini Definiti e Riferimenti **675**
17.1 Definizioni e Acronimi . . . 675

17.1.1 Acronimi . . . 688
17.2 Linguaggio Normativo e Convenzioni . . . 689
17.3 Riferimenti Normativi . . . 690
17.3.1 Risorse Ufficiali . . . 690
17.4 Technical References . . . 692
17.4.1 Wallet Architecture and Reference Frameworks . . . 692
17.4.2 EUDI Wallet Standards and Technical Specifications . . . 692
17.4.3 Infrastructure of Trust . . . 693
17.4.4 Digital Credential Data Format . . . 693
17.4.5 Digital Credential Issuance . . . 693
17.4.6 Digital Credential Presentation . . . 694
17.4.7 Digital Credential Revocation Check Mechanisms . . . 694
17.4.8 National Data Interoperability Platform Specifications . . . 694
17.4.9 National Digital Identity Platform Specifications . . . 694
17.4.10 Security and Protection Profiles . . . 695

18 Come contribuire **697**
18.1 Ringraziamenti . . . 698

19 Rilasci Open Source **701**
19.1 1. Principi Generali . . . 701
19.2 2. Requisiti Standard . . . 701
19.2.1 2.1 Licenze . . . 702
19.2.2 2.2 Gestione del Codice e Trasparenza . . . 702
19.2.3 2.3 Sicurezza e Qualità . . . 702
19.3 3. Fornitori di Wallet . . . 703
19.3.1 3.1 Open Source Obbligatorio . . . 703
19.4 4. Fornitori di Credenziali e Relying Party . . . 703
19.4.1 4.1 Componenti di integrazione . . . 703
19.4.2 4.2 Infrastruttura di backend . . . 703
19.5 5. Divulgazione Responsabile e Gestione delle Vulnerabilità . . . 703

20 Appendice **705**
20.1 Istanza dell'Applicazione Mobile . . . 705
20.1.1 Inizializzazione dell'Istanza dell'Applicazione Mobile . . . 705
20.2 e-Service PDND . . . 714
20.2.1 Requisiti e Pattern di Sicurezza . . . 715
20.2.2 Emissione del Voucher PDND . . . 717
20.2.3 Reperimento delle Chiavi . . . 732
20.2.4 Utilizzo dell'e-Service . . . 738
20.3 PDND e-Service Template . . . 748
20.3.1 Definizione e linee guida dell'e-service template PDND . . . 749

20.4 Test Plans 767
20.4.1 Struttura della Matrice di Test 768
20.5 Specifica OpenAPI del Fornitore di Wallet PDND 879

Indice **889**

Questo documento fornisce l'architettura tecnica, il framework di implementazione e i requisiti di progettazione da adottare per le Soluzioni Tecniche del Sistema IT-Wallet.

CAPITOLO 1

Introduzione

Nell'ultimo decennio, la digitalizzazione ha trasformato radicalmente il modo in cui cittadini e imprese interagiscono con i servizi pubblici e privati, introducendo nuove forme di accesso ai servizi sicure, accessibili e user-friendly.

In Italia, il Decreto-Legge n. 19 del 2 marzo 2024, convertito, con modificazioni, dalla Legge n. 56 del 29 aprile 2024, ha introdotto l'articolo 64-quater del Decreto Legislativo n. 82 del 7 marzo 2005, istituendo il Sistema di Portafoglio Digitale Italiano - Sistema IT-Wallet. Il Sistema IT-Wallet consente a persone fisiche o giuridiche di accedere a servizi pubblici e privati attraverso la presentazione sicura di Attestati Elettronici, attestanti diritti, deleghe, caratteristiche, licenze o qualifiche. L'articolo 64-quater prevede inoltre l'adozione di uno o più decreti attuativi per definire le regole che disciplinano il funzionamento del Sistema IT-Wallet, compresi i ruoli delle entità coinvolte, i requisiti tecnici e di sicurezza, e i principi di sostenibilità economica, di cui queste Specifiche Tecniche – redatte attraverso un processo aperto e collaborativo – costituiscono parte integrante.

Grazie al Sistema IT-Wallet, le persone fisiche e giuridiche possono fornire direttamente, tramite il proprio Wallet, le informazioni necessarie per accedere ai servizi offerti da enti pubblici e privati sotto forma di Attestati Elettronici. Analogamente a un portafoglio fisico, un'Istanza del Wallet può contenere dati relativi all'identità o ai documenti, come la patente di guida o la tessera sanitaria, nonché una vasta gamma di informazioni digitali verificabili, come una qualifica professionale, un diploma di istruzione, una licenza o un attributo certificato.

I principali ruoli nell'ecosistema Wallet sono elencati di seguito:

- **Fornitore di Attestati Elettronici**: soggetti che rilasciano Attestati Elettronici agli Utenti;

- **Verificatori di Attestati Elettronici**: soggetti che richiedono all'Utente presentazioni di Attestati Elettronici, per finalità di Autenticazione e autorizzazione;
- **Utenti**: individui che possiedono un'Istanza del Wallet e hanno il controllo sugli Attestati Elettronici che possono richiedere, acquisire, memorizzare e presentare alle Verificatori di Attestati Elettronici;

In questo modello, il Fornitore di Attestati Elettronici (ad esempio, un'istituzione educativa) fornisce Attestati Elettronici all'Utente, che può memorizzarle nella propria Istanza del Wallet. L'Istanza del Wallet è tipicamente fornita come applicazione mobile sullo smartphone dell'Utente.

Ciò che distingue questo nuovo approccio dai precedenti sistemi di gestione dell'accesso all'identità è che gli Attestati Elettronici si riferiscono a caratteristiche, qualità o proprietà, già autenticate alla fonte. Questi Attestati Elettronici possono essere utilizzati dall'Utente senza che i Fornitori di Attestati Elettronici siano a conoscenza del loro utilizzo. Durante l'uso degli Attestati Elettronici, nessuna informazione sull'utilizzo viene rilasciata a terze parti poiché la relazione è esclusiva tra l'Utente e il Verificatore di Attestati Elettronici, in modo trasparente e informato. Lo sviluppo del Sistema IT-Wallet include un processo di sperimentazione graduale, finalizzato a testare il Wallet e valutarne l'impatto in contesti reali. Questo processo è progettato per validare componenti tecniche, elementi di esperienza utente e meccanismi di interoperabilità, garantendo al contempo un'adozione progressiva e controllata del Sistema. Inoltre, supporta il miglioramento continuo del Sistema IT-Wallet e il suo graduale allineamento con il Portafoglio Europeo di Identità Digitale (EUDI Wallet), sia in termini di architettura che di conformità con le specifiche europee in evoluzione.

Altri elementi chiave che caratterizzano questo nuovo paradigma di Portafoglio di Identità Digitale includono:

- **Riservatezza e controllo**: i Wallet consentono agli individui di mantenere il controllo sulle informazioni fornite all'interno degli Attestati Elettronici presentate. Possono scegliere quali attributi o Attestati Elettronici presentare e a chi;
- **Sicurezza**: i Wallet sfruttano meccanismi crittografici per l'integrità e la sicurezza dei dati scambiati. Ciò evita il furto di identità, le frodi e gli accessi non autorizzati;
- **Interoperabilità**: i Wallet promuovono l'interoperabilità consentendo a diversi sistemi e organizzazioni di riconoscere e verificare le identità, abilitando interazioni affidabili tra individui, organizzazioni e persino oltre i confini;
- **Efficienza e riduzione dei costi**: gli individui possono gestire facilmente i propri Attestati Elettronici, evitare di gestire più token di identità e ridurre i processi ripetitivi di verifica dell'identità.

1.1 Ambito

Queste Specifiche Tecniche sono destinate a completare le Linee Guida previste dall'articolo 64-quater del Decreto Legislativo n. 82/2005 (CAD). Sia le Linee Guida che queste Specifiche Tecniche, una volta formalmente adottate, diventeranno parte del quadro normativo per il Sistema IT-Wallet. Saranno periodicamente aggiornate, ove necessario, alla luce dei risultati della fase di sperimentazione, dell'adozione di nuovi atti legislativi nazionali o europei e dei requisiti in evoluzione in termini di sicurezza e interoperabilità. Queste Specifiche Tecniche perseguono due obiettivi principali e rappresentano una componente fondamentale del quadro di implementazione per il Sistema IT-Wallet.

Il primo obiettivo è quello di fornire un insieme chiaro e strutturato di raccomandazioni, risorse e requisiti di progettazione relativi agli elementi del Sistema IT-Wallet che impattano sull'Esperienza dell'Utente. Il documento, distinguendo tra aspetti normativi obbligatori e buone pratiche progettuali e implementative, mira a fornire agli enti pubblici e privati interessati a partecipare al Sistema IT-Wallet quanto necessario per:

- facilitare la comprensione e l'adozione del Modello di Servizio, aumentando il numero di potenziali servizi e opportunità di utilizzo per l'Utente;
- adottare l'Identità Visiva del Sistema IT-Wallet al fine di migliorarne l'affidabilità e la riconoscibilità per l'Utente;
- garantire la coerenza di progettazione tra macro-funzionalità e singole interazioni tra l'Utente e i Touchpoint del servizio;
- mantenere un adeguato livello di qualità, promuovendo i principi di usabilità, accessibilità e inclusività.

Il secondo focus è definire l'architettura tecnica e il quadro di riferimento che servirà da linea guida per tutte le parti coinvolte nello sviluppo delle Soluzioni Tecniche del Sistema IT-Wallet.

La documentazione, gli strumenti e le risorse aggiuntive - di seguito definite Risorse Ufficiali - per la progettazione e lo sviluppo delle Soluzioni Tecniche del Sistema IT-Wallet saranno presto disponibili nella sezione official-resources:Risorse Ufficiali: e saranno aggiornate periodicamente.

1.2 Linguaggio Normativo e Convenzioni

Le parole chiave "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" e "OPTIONAL" in questo documento devono essere interpretate come descritto in BCP 14 [RFC2119] [RFC8174] quando, e solo quando, appaiono in maiuscolo, come mostrato qui.

1.3 Come leggere le Specifiche

Questa specifica è progettata per soddisfare i requisiti di molteplici stakeholder all'interno del Sistema IT-Wallet. Ogni ruolo ha responsabilità e ambiti diversi e diverse esigenze informative. Questa sezione fornisce percorsi di lettura personalizzati per aiutare il lettore a navigare in modo efficiente verso i contenuti più rilevanti per gli obiettivi di implementazione.

1.3.1 Panoramica della Struttura delle Specifiche

La specifica è organizzata nelle seguenti sezioni principali:

- **Sezione** *Introduzione* (pagina 3): Stabilisce l'ambito e il linguaggio normativo per l'ecosistema IT-Wallet.
- **Sezione** *Panoramica dell'Architettura* (pagina 21): Fornisce una visione di alto livello dell'Architettura, in termini di governance e processi operativi abilitati.
- **Sezione** *Brand Identity* (pagina 15): Fornisce i requisiti relativi alla Brand Identity del Sistema IT-Wallet, le indicazioni relative al naming e all'applicazione degli elementi visivi che identificano l'ecosistema.
- **Sezione** *Design dell'Esperienza Utente* (pagina 39): Fornisce i principi di design e i requisiti funzionali di alto livello per garantire un'Esperienza Utente di qualità in tutte le fasi di interazione tra l'Utente e il servizio.
- **Sezione** *L'Infrastruttura di Trust* (pagina 117): Definisce il modello di trust basato sulla federazione, le relazioni tra entità e i meccanismi di valutazione della fiducia che proteggono l'intero ecosistema.
- **Sezione** *Entità* (pagina 287): Requisiti di implementazione completi per ogni partecipante all'ecosistema: Soluzioni Wallet, Fornitori di Credenziali, Relying Party e Fonti Autentiche, inclusi i loro componenti, modelli di interazione e requisiti di configurazione.
- **Sezione** *Gestione degli Attestati Elettronici* (pagina 379): Copre i modelli di dati e i formati delle Credenziali Elettroniche, la gestione del ciclo di vita, la verifica della validità e la struttura del catalogo delle Credenziali.
- **Sezione** *Flussi relativi agli Attestati Elettronici* (pagina 443): Guida dettagliata all'implementazione per i flussi di emissione e presentazione delle Credenziali Elettroniche, inclusi i flussi di interazione remota e di prossimità, e l'identity matching dopo la presentazione.
- **Sezione** *Endpoints* (pagina 571): Specifiche tecniche per tutti gli endpoint API esposti da ciascun tipo di entità, inclusi gli endpoint di federazione e le integrazioni di servizi specializzati.

- **Sezione** *Algoritmi Crittografici* (pagina 647), *Considerazioni di Sicurezza e Privacy* (pagina 653), e *Politiche Generali di Conservazione dei Log* (pagina 671) (**Supporto all'Implementazione**): Requisiti crittografici, considerazioni sulla sicurezza e sulla privacy, e politiche di conservazione dei log essenziali per implementazioni conformi.
- **Sezione** *Termini Definiti e Riferimenti* (pagina 675), official-resources:Risorse Ufficiali, *Come contribuire* (pagina 697), e *Rilasci Open Source* (pagina 701) (**Terminologia e Riferimenti**): Terminologia completa, riferimenti normativi, documentazione, risorse e strumenti aggiuntivi, linee guida per i contributi.
- **Sezione** *Appendice* (pagina 705): Fornisce dettagli tecnici supplementari, modelli di implementazione e framework di test, inclusa la gestione delle istanze di applicazioni mobili, specifiche di integrazione della piattaforma nazionale e matrici di test complete per la validazione dell'ecosistema.

1.3.2 Percorsi di lettura per ruolo

Per facilitare la consultazione degli argomenti trattati in queste specifiche, di seguito sono riportati percorsi di lettura organizzati per ruolo. Ogni percorso è strutturato in base alle diverse fasi di adesione e partecipazione al Sistema IT-Wallet e contiene riferimenti diretti alle sezioni più rilevanti per ciascun ruolo.

I percorsi di lettura proposti hanno finalità orientativa e non sostituiscono la necessità di consultare le altre sezioni quando è richiesta una comprensione più ampia del sistema.

Per le entità interessate a operare in più ruoli, si raccomanda di approfondire tutti i percorsi di lettura relativi ai ruoli di interesse.

Fonte Autentica

La Fonte Autentica si occupa di rendere disponibili i dati affidabili alla base dei (Q)EAA e di garantirne e gestirne in modo sicuro l'assoluta accuratezza e integrità.

Fase 1: Scoperta

Comprendere il funzionamento generale dell'ecosistema, l'architettura tecnica e l'Infrastruttura di Trust.

- **Sezione** *Introduzione* (pagina 3): Scopo e contesto normativo del Sistema IT-Wallet.
- **Sezione** *Panoramica dell'Architettura* (pagina 21): Panoramica dell'architettura del Sistema IT-Wallet in termini di governance e processi operativi abilitati.
- **Sezione** *L'Infrastruttura di Trust* (pagina 117): Requisiti chiave del modello di fiducia federato e dei meccanismi di valutazione della fiducia tra le entità.

- **Sezione** *Termini Definiti e Riferimenti* (pagina 675): Terminologia completa, riferimenti normativi, documentazione, risorse e strumenti aggiuntivi, linee guida per i contributi.

Fase 2: Progettazione

Comprendere i requisiti e progettare gli aspetti, le funzionalità e le caratteristiche specifiche del (Q)EAA da emettere.

- **Sezione** *Design dell'Esperienza Utente* (pagina 39): Requisiti chiave relativi alle modalità con cui consentire agli Utenti di ottenere i (Q)EAA, alla struttura dei (Q)EAA, al loro stato e alla loro gestione nel tempo.
- **Sezione** *Gestione degli Attestati Elettronici* (pagina 379): Requisiti tecnici e funzionali relativi al ciclo di vita dei (Q)EAA.
- **Sezione** *PDND e-Service Template* (pagina 748): Modello di riferimento contenente tutti i metadati tecnici e descrittivi necessari per la definizione dell'e-service.

Fase 3: Implementazione

Implementare le interfacce tecnologiche necessarie per comunicare con il Fornitore di Attestati Elettronici e gestire l'intero ciclo di vita dei dati associati ai (Q)EAA.

- **Sezione** *Fonti Autentiche* (pagina 376): Ruolo e responsabilità della Fonte Autentica.
- **Sezione** *e-Service PDND* (pagina 714): Specifiche obbligatorie per l'integrazione con la PDND (Piattaforma Digitale Nazionale Dati) e relativi requisiti di interoperabilità per la pubblicazione di un e-service.
- **Sezione** *Endpoint delle Fonti Autentiche* (pagina 634): Requisiti chiave per l'implementazione delle API che consentono al Credential Issuer di recuperare in modo sicuro e coerente i dati autorevoli e di gestirne il ciclo di vita tramite gli endpoint del Signal Hub.
- **Sezione** *Infrastruttura del Registro* (pagina 159): Approfondimento sui componenti del Registro di interesse per la Fonte Autentica.
- **Sezione** *Politiche Generali di Conservazione dei Log* (pagina 671): Requisiti generali per la conservazione dei log e requisiti specifici per le Fonti Autentiche, in conformità alla norma ISO/IEC 27001.
- **Sezione** *Test Plans* (pagina 767): Guida alla configurazione dell'ambiente di test e alla validazione delle interazioni backend tramite l'utilizzo delle matrici di test fornite dall'ecosistema.

Fase 4: Registrazione

Registrarsi al sistema come Fonte Autentica, completando le procedure amministrative e tecniche richieste.

- **Sezione** *Sistema di Onboarding* (pagina 25): Panoramica dell'architettura del sistema di onboarding e del processo di registrazione della Fonte Autentica.
- **Sezione** *Onboarding delle Entità* (pagina 239): Approfondimento sulle procedure tecniche per la registrazione della Fonte Autentica.
- **Sezione** *Operazioni di Gestione dei Certificati X.509* (pagina 271): Procedure operative per la gestione dei certificati X.509 nell'ambito della federazione IT-Wallet.

Fornitore di Wallet

Il Fornitore di Wallet si occupa della progettazione e dello sviluppo della Soluzione Wallet, che consente all'Utente di conservare, gestire e presentare il proprio PID e i propri (Q)EAA.

Fase 1: Scoperta

Comprendere il funzionamento generale dell'ecosistema, l'architettura tecnica e l'Infrastruttura di Trust.

- **Sezione** *Introduzione* (pagina 3): Scopo e contesto normativo del Sistema IT-Wallet.
- **Sezione** *Panoramica dell'Architettura* (pagina 21): Panoramica dell'architettura del Sistema IT-Wallet in termini di governance e processi operativi abilitati.
- **Sezione** *L'Infrastruttura di Trust* (pagina 117): Requisiti chiave del modello di fiducia federato e dei meccanismi di valutazione della fiducia tra le entità.
- **Sezione** *Termini Definiti e Riferimenti* (pagina 675): Terminologia completa, riferimenti normativi, documentazione, risorse e strumenti aggiuntivi, linee guida per i contributi.

Fase 2: Progettazione

Comprendere i requisiti relativi all'Esperienza Utente e progettare la Soluzione Wallet secondo modelli comuni, al fine di garantirne l'usabilità e l'accessibilità.

- **Sezione** *Brand Identity* (pagina 15): Panoramica della Brand Identity del Sistema IT-Wallet e indicazioni sugli asset di riferimento per il Fornitore di Wallet.
- **Sezione** *Design dell'Esperienza Utente* (pagina 39): Requisiti chiave relativi ai modelli di interazione, ai layout delle interfacce e agli asset grafici, al fine di garantire un'esperienza utente efficace, fluida e assicurare coerenza tra le diverse Soluzioni Wallet.
- **Sezione** *Gestione degli Attestati Elettronici* (pagina 379): Requisiti tecnici e funzionali relativi al ciclo di vita dei (Q)EAA.

Fase 3: Implementazione

Implementare la Soluzione Wallet in conformità a specifici standard tecnologici, al fine di garantire la comunicazione tra la Soluzione Wallet e gli altri attori del Sistema.

- **Sezione** *Soluzione Wallet* (pagina 287): Requisiti tecnici e funzionali relativi ai componenti, alle funzionalità e al ciclo di vita necessari per la configurazione della Soluzione Wallet.
- **Sezione** *Flussi relativi agli Attestati Elettronici* (pagina 443): Requisiti tecnici e funzionali relativi ai flussi di emissione e presentazione dei (Q)EAA.
- **Sezione** *Endpoint del Fornitore di Wallet* (pagina 571): Requisiti chiave per l'implementazione delle interfacce (API) del Fornitore di Wallet necessarie a garantire l'interoperabilità tra le entità.
- **Sezione** *Infrastruttura del Registro* (pagina 159): Approfondimento sui componenti del Registro.
- **Sezione** *Algoritmi Crittografici* (pagina 647): Selezione e implementazione degli standard crittografici necessari a garantire la sicurezza delle chiavi e delle transazioni.
- **Sezione** *Considerazioni di Sicurezza e Privacy* (pagina 653): Requisiti di sicurezza e conformità per le Soluzioni Wallet.
- **Sezione** *Politiche Generali di Conservazione dei Log* (pagina 671): Requisiti generali per la conservazione dei log e requisiti specifici per i Fornitori di Wallet, in conformità alla norma ISO/IEC 27001.
- **Sezione** *Istanza dell'Applicazione Mobile* (pagina 705): Requisiti relativi all'istanza dell'applicazione mobile, con riferimento alla richiesta e alla risposta di inizializzazione.
- **Sezione** *Test Plans* (pagina 767): Guida alla configurazione dell'ambiente di test e alla validazione delle interazioni backend tramite l'utilizzo delle matrici di test fornite dall'ecosistema.

Fase 4: Registrazione

Registrarsi al sistema come Fornitore di Wallet, completando le procedure amministrative e tecniche richieste per il riconoscimento della Soluzione Wallet da parte del sistema.

- **Sezione** *Sistema di Onboarding* (pagina 25): Panoramica dell'architettura del sistema di onboarding e del processo di registrazione del Fornitore di Wallet.
- **Sezione** *Onboarding delle Entità* (pagina 239): Approfondimento sulle procedure tecniche per la registrazione del Fornitore di Wallet.
- **Sezione** *Operazioni di Gestione dei Certificati X.509* (pagina 271): Procedure operative per la gestione dei certificati X.509 nell'ambito della federazione IT-Wallet.

Fornitore di Attestati Elettronici

Il Fornitore di Attestati Elettronici si occupa di trasformare i dati autorevoli di origine provenienti dalle Fonti Autentiche in (Q)EAA e di gestirne l'intero ciclo di vita, dall'emissione alla revoca o alla scadenza.

Fase 1: Scoperta

Comprendere il funzionamento generale dell'ecosistema, l'architettura tecnica e l'Infrastruttura di Trust.

- **Sezione** *Introduzione* (pagina 3): Scopo e contesto normativo del Sistema IT-Wallet.
- **Sezione** *Panoramica dell'Architettura* (pagina 21): Panoramica dell'architettura del Sistema IT-Wallet in termini di governance e processi operativi abilitati.
- **Sezione** *L'Infrastruttura di Trust* (pagina 117): Requisiti chiave del modello di fiducia federato e dei meccanismi di valutazione della fiducia tra le entità.
- **Sezione** *Termini Definiti e Riferimenti* (pagina 675): Terminologia completa, riferimenti normativi, documentazione, risorse e strumenti aggiuntivi, linee guida per i contributi.

Fase 2: Progettazione

Comprendere i requisiti e progettare tecnicamente i (Q)EAA, strutturandone i metadati in conformità ai requisiti tecnici e normativi.

- **Sezione** *Design dell'Esperienza Utente* (pagina 39): Requisiti chiave relativi alla struttura, all'emissione, allo stato e alla gestione nel tempo dei (Q)EAA.
- **Sezione** *Gestione degli Attestati Elettronici* (pagina 379): Requisiti tecnici e funzionali relativi al ciclo di vita dei (Q)EAA.

Fase 3: Implementazione

Sviluppare endpoint secondo specifici protocolli e implementare le funzionalità di emissione, rinnovo, revoca e tutte le altre funzioni tecniche di gestione del ciclo di vita dei (Q)EAA.

- **Sezione** *Soluzione del Fornitore di Attestati Elettronici* (pagina 330): Requisiti tecnici e funzionali relativi ai componenti e ai modelli di interazione necessari per l'emissione e la gestione del ciclo di vita dei (Q)EAA.
- **Sezione** *Flussi relativi agli Attestati Elettronici* (pagina 443): Requisiti tecnici e funzionali relativi ai flussi di emissione e presentazione dei (Q)EAA.
- **Sezione** *Endpoint del Credential Issuer* (pagina 592): Requisiti chiave per l'implementazione dei metadati del Fornitore di Attestati Elettronici e degli endpoint di autorizzazione.
- **Sezione** *Algoritmi Crittografici* (pagina 647): Selezione e implementazione degli standard crittografici necessari a garantire la sicurezza delle chiavi e delle transazioni.

- **Sezione** *e-Service PDND* (pagina 714): Specifiche obbligatorie per l'integrazione con la PDND (Piattaforma Digitale Nazionale Dati) e relativi requisiti di interoperabilità per la pubblicazione di un e-service.
- **Sezione** *Considerazioni di Sicurezza e Privacy* (pagina 653): Requisiti di sicurezza e conformità per le soluzioni implementate.
- **Sezione** *Politiche Generali di Conservazione dei Log* (pagina 671): Requisiti generali per la conservazione dei log e requisiti specifici per i Fornitori di Attestati Elettronici, in conformità alla norma ISO/IEC 27001.
- **Sezione** *Test Plans* (pagina 767): Guida alla configurazione dell'ambiente di test e alla validazione delle interazioni backend tramite l'utilizzo delle matrici di test fornite dall'ecosistema.

Fase 4: Registrazione

Registrarsi al sistema come Credential Issuer, completando le procedure amministrative e tecniche previste affinché i (Q)EAA emessi verso i Wallet siano riconosciuti come affidabili dal Sistema.

- **Sezione** *Sistema di Onboarding* (pagina 25): Panoramica dell'architettura del sistema di onboarding e del processo di registrazione del Fornitore di Attestati Elettronici.
- **Sezione** *Onboarding delle Entità* (pagina 239): Approfondimento sulle procedure tecniche per la registrazione del Fornitore di Attestati Elettronici.
- **Sezione** *Operazioni di Gestione dei Certificati X.509* (pagina 271): Procedure operative per la gestione dei certificati X.509 nell'ambito della federazione IT-Wallet.

Fornitori di servizi (Relying Party)

I Fornitori di servizi (Relying Party) si occupano di richiedere, ricevere e verificare in modo sicuro l'autenticità e la validità del PID e dei (Q)EAA presentati dall'Utente per consentire l'accesso ai servizi online e offline.

Fase 1: Scoperta

Comprendere il funzionamento generale dell'ecosistema, l'architettura tecnica e l'Infrastruttura di Trust.

- **Sezione** *Introduzione* (pagina 3): Scopo e contesto normativo del Sistema IT-Wallet.
- **Sezione** *Panoramica dell'Architettura* (pagina 21): Panoramica dell'architettura del Sistema IT-Wallet in termini di governance e processi operativi abilitati.
- **Sezione** *L'Infrastruttura di Trust* (pagina 117): Requisiti chiave del modello di fiducia federato e dei meccanismi di valutazione della fiducia tra le entità.

- **Sezione** *Termini Definiti e Riferimenti* (pagina 675): Terminologia completa, riferimenti normativi, documentazione, risorse e strumenti aggiuntivi, linee guida per i contributi.

Fase 2: Progettazione

Comprendere i requisiti di Esperienza Utente e progettare le funzionalità di verifica necessarie all'erogazione del servizio all'Utente finale.

- **Sezione** *Brand Identity* (pagina 15): Panoramica della Brand Identity del Sistema IT-Wallet e indicazioni sugli asset da adottare da parte della Relying Party.
- **Sezione** *Design dell'Esperienza Utente* (pagina 39): Requisiti chiave relativi ai modelli di interazione, ai layout delle interfacce e agli asset grafici, al fine di garantire un'esperienza utente efficace, fluida e assicurare coerenza tra i sistemi di presentazione e verifica.
- **Sezione** *Gestione degli Attestati Elettronici* (pagina 379): Requisiti tecnici e funzionali relativi al ciclo di vita dei (Q)EAA.

Fase 3: Implementazione

Implementare le funzionalità di verifica secondo specifici protocolli, al fine di inviare richieste di verifica al Wallet e ricevere una risposta con l'autorizzazione dell'Utente.

- **Sezione** *Soluzione di Relying Party* (pagina 361): Requisiti tecnici e funzionali relativi ai componenti e alle funzionalità per la verifica del PID e dei (Q)EAA.
- **Sezione** *Flussi relativi agli Attestati Elettronici* (pagina 443): Requisiti tecnici e funzionali relativi ai flussi di emissione e presentazione dei (Q)EAA.
- **Sezione** *Identity Matching e Identity Reconciliation* (pagina 567): Regole di identity matching e identity reconciliation dopo la presentazione degli attributi di identificazione personale.
- **Sezione** *Endpoint della Relying Party* (pagina 624): Requisiti chiave per l'implementazione degli endpoint di verifica dei (Q)EAA.
- **Sezione** *Considerazioni di Sicurezza e Privacy* (pagina 653): Requisiti di sicurezza e conformità per le soluzioni implementate.
- **Sezione** *Politiche Generali di Conservazione dei Log* (pagina 671): Requisiti generali per la conservazione dei log e requisiti specifici per le Relying Party, in conformità alla norma ISO/IEC 27001.
- **Sezione** *Test Plans* (pagina 767): Guida alla configurazione dell'ambiente di test e alla validazione delle interazioni backend tramite l'utilizzo delle matrici di test fornite dall'ecosistema.

Fase 4: Registrazione

Registrarsi al sistema come Relying Party, completando le procedure amministrative e tecniche previste, e diventare soggetto affidabile per la richiesta dei dati degli Utenti.

- **Sezione** *Sistema di Onboarding* (pagina 25): Panoramica dell'architettura del sistema di onboarding e del processo di registrazione della Relying Party.
- **Sezione** *Onboarding delle Entità* (pagina 239): Approfondimento sulle procedure tecniche per la registrazione della Relying Party.
- **Sezione** *Operazioni di Gestione dei Certificati X.509* (pagina 271): Procedure operative per la gestione dei certificati X.509 nell'ambito della federazione IT-Wallet.

CAPITOLO 2

Brand Identity

La Brand Identity del Sistema IT-Wallet definisce la personalità del sistema e si declina in una serie di elementi codificati che la contraddistinguono, tra cui il naming e gli elementi propri dell'Identità Visiva. Questa contribuisce e valorizza il Modello della Fiducia alla base dell'intero ecosistema, introducendo elementi visivi riconoscibili.

È importante che la consapevolezza dell'Utente e la sua fiducia nell'ecosistema siano favoriti dalla presenza di elementi grafici, da integrare nelle Soluzioni Tecniche e nei diversi Touchpoint.

Per garantire quindi che la presenza della Brand Identity nelle diverse Soluzioni Tecniche sia uniforme, coerente e favorisca un'Esperienza Utente di qualità, gli Attori Primari DEVONO rispettare i requisiti di applicazione della Brand Identity del Sistema IT-Wallet nelle loro Soluzioni Tecniche. In particolare, tutti gli Attori Primari:

- DEVONO seguire esclusivamente le indicazioni specificate nel Brand Manual, indicato nella sezione official-resources:Risorse Ufficiali;
- DEVONO utilizzare esclusivamente le Risorse Ufficiali che sono disponibili nella relativa sezione official-resources:Risorse Ufficiali di queste Specifiche Tecniche.

2.1 Naming

Il termine ”Sistema IT-Wallet”, così come definito dalla norma istitutiva, rappresenta la denominazione ufficiale che DOVREBBE essere utilizzata in contesti a carattere istituzionale, formale o giuridico per riferirsi al sistema.

La declinazione contratta ”IT-Wallet” rappresenta, invece, il Brand naming ufficiale che DEVE essere utilizzato in ogni contesto comunicativo e divulgativo, scritto o verbale, fisico o digitale, per riferirsi al ”Sistema IT-Wallet”.

Di seguito sono riportati i requisiti di scrittura e pronuncia. Tutti coloro che intendono riferirsi al Sistema IT-Wallet:

- DEVONO rispettare l'uso delle maiuscole nel termine ”IT” e nelle iniziali di ”Wallet” e ”Sistema”;
- DEVONO rispettare l'uso del trattino tra i termini “IT” e “Wallet”, senza l'uso di spaziature;
- DEVONO rispettare la corretta pronuncia del termine “IT”, separando foneticamente le due lettere “I” e “T”. In italiano la fonetica corretta è /i ti/ e non /it/, mentre in inglese la fonetica corretta è /ˌaɪ ˈtiː/ e non /ɪt/, /aɪt/ o /ɪti/.

2.2 Identità Visiva

Il Brand ”IT-Wallet” ha una propria Identità Visiva. I partecipanti dell'ecosistema DEVONO applicarla per dare visibilità della propria partecipazione al Sistema IT-Wallet, ma NON DEVONO utilizzare l'Identità Visiva del Brand “IT-Wallet” in sostituzione dell'Identità Visiva della propria organizzazione o la propria Soluzione Tecnica.

Per garantire una corretta coesistenza tra l'Identità Visiva del Brand “IT-Wallet” e l'Identità Visiva della propria organizzazione o delle proprie Soluzioni Tecniche, gli Attori Primari DEVONO utilizzare esclusivamente le Risorse Ufficiali e rispettare le indicazioni del Brand Manual.

2.2.1 Logo

Il Logo è l'elemento grafico ufficiale che permette l'immediata riconoscibilità del Sistema IT-Wallet. È composto da un pittogramma e dal logotipo “IT-Wallet”.

Di seguito sono riportati i requisiti generali di utilizzo del Logo. Tutti gli attori che intendono riferirsi al Sistema:

- POSSONO utilizzare il Logo per riferirsi al Sistema IT-Wallet ma NON DEVONO utilizzare il Logo per identificare una specifica Soluzione Tecnica;



Fig. 2.1: Logo del Brand “IT-Wallet” su sfondo chiaro



Fig. 2.2: Pittogramma del Brand “IT-Wallet” su sfondo chiaro

- DEVONO utilizzare il Logo reso disponibile nelle Risorse Ufficiali e DEVONO rispettare tutte le relative specifiche di utilizzo fornite;
- DEVONO utilizzare il Logo in formato `application/svg+xml`;
- NON DEVONO alterare, distorcere, modificare il Logo o sostituirlo con loghi non ufficiali;
- DEVONO utilizzare il Logo garantendo l’area di rispetto minima stabilita nel Brand Manual, al fine di garantirne un’adeguata visibilità e riconoscibilità. NON DEVONO permettere che altri elementi grafici o testuali interferiscano con questa area di rispetto;
- NON DEVONO ridimensionare il Logo oltre i limiti minimi stabiliti nel Brand Manual, in modo da garantire sempre una leggibilità ottimale su qualsiasi formato o dispositivo;
- NON DEVONO utilizzare il Logo su sfondi di colore che ne compromettano la visibilità o la leggibilità. DEVONO garantire un contrasto adeguato tra il Logo e lo sfondo, in conformità con quanto definito nel Brand Manual;
- POSSONO utilizzare il Logo in associazione a loghi, marchi o simboli di altri attori del sistema, purchè in accordo con quando stabilito nel Brand Manual;
- POSSONO utilizzare il pittogramma del Logo come versione sintetica del Logo, in accordo con quando stabilito nel Brand Manual;

- POSSONO utilizzare il pittogramma del Logo all'interno di una forma o di un pulsante per rendere evidente il richiamo a IT-Wallet, come per esempio negli Engagement Buttons (vedi *Authentication Button* (pagina 96)), in accordo con quando stabilito nel Brand Manual.

Per approfondimenti sul Logo e relativo pittogramma, consultare il Brand Manual, indicato nella sezione official-resources:Risorse Ufficiali. Le Risorse Ufficiali del Logo sono disponibili nella sezione official-resources:Risorse Ufficiali di queste Specifiche Tecniche.

2.2.2 Trust Mark

Il Trust Mark è l'elemento grafico ufficiale che dà prova all'Utente dell'appartenenza degli Attori Primari e delle relative Soluzioni Tecniche, al Sistema IT-Wallet.



Fig. 2.3: Trust Mark su sfondo chiaro

Di seguito sono riportati i requisiti di utilizzo del Trust Mark, validi sia in riferimento a contesti di utilizzo fisici che digitali (e.g. siti web, applicazioni, documenti cartacei, materiali informativi stampati o video, etc.). Tutti gli Attori Primari:

- DEVONO utilizzare il Trust Mark esclusivamente per dare prova dell'appartenenza al Sistema IT-Wallet e non DEVONO utilizzarlo per scopi diversi;
- DEVONO esporre il Trust Mark esclusivamente nelle Soluzioni Tecniche che hanno concluso con successo il processo di registrazione;
- DEVONO esporre il Trust Mark in maniera chiara e visibile all'Utente fin dal primo accesso alla Soluzione Tecnica e DEVONO renderlo consultabile all'interno della Soluzione Tecnica;
- DEVONO utilizzare esclusivamente il Trust Mark reso disponibile nelle Risorse Ufficiali e DEVONO rispettare tutte le relative specifiche di utilizzo fornite nel official-resources: Brand Manual;
- NON DEVONO alterare, distorcere, modificare il Trust Mark o sostituirlo con elementi non ufficiali;
- NON DEVONO ridimensionare il Trust Mark oltre i limiti minimi stabiliti dal official-resources:Brand Manual, in modo da garantire sempre una leggibilità ottimale su qualsiasi formato o dispositivo;

- DEVONO utilizzare il Trust Mark garantendo l'area di rispetto minima definita nel official-resources:Brand Manual, al fine di garantirne un'adeguata visibilità e riconoscibilità. NON DEVONO permettere ad altri elementi grafici o testuali di interferire con questa area di rispetto;
- NON DEVONO utilizzare il Trust Mark su sfondi di colore che ne compromettano la visibilità o la leggibilità e DEVONO garantire un contrasto adeguato tra il Trust Mark e lo sfondo, in conformità con quanto definito nel official-resources:Brand Manual;
- POSSONO associare il Trust Mark a loghi, marchi o simboli di altri attori del sistema, purchè in accordo con quando stabilito nel Brand Manual.

Per approfondimenti sul Trust Mark, consultare il Brand Manual, indicato nella sezione official-resources:Risorse Ufficiali. La Risorsa Ufficiale del Trust Mark è disponibile nella sezione official-resources:Risorse Ufficiali di queste specifiche tecniche.

CAPITOLO 3

Panoramica dell'Architettura

Il Sistema IT-Wallet è un ecosistema federato che consente la gestione sicura dell'Identità Digitale e lo scambio di Attestati Elettronici per cittadini e organizzazioni. L'ecosistema IT-Wallet è costruito su un'architettura multi-livello, dove gli organi di governance stabiliscono e mantengono l'infrastruttura di trust, gli Attori Primari implementano e gestiscono le Soluzioni Tecniche, e i sistemi esterni forniscono servizi aggiuntivi.

I seguenti diagrammi illustrano la panoramica dell'architettura IT-Wallet.

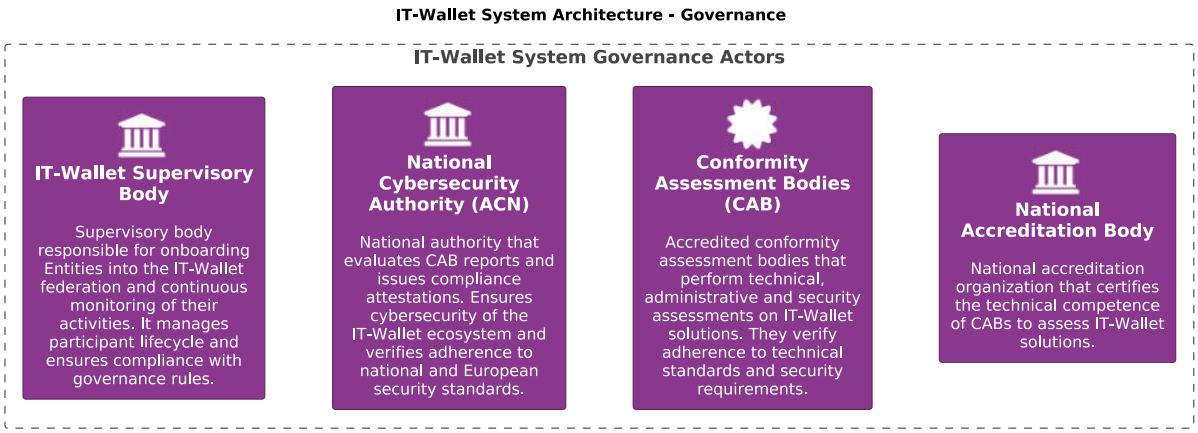


Fig. 3.1: Panoramica dell'architettura IT-Wallet - Governance.

Il livello di governance richiede a tutti i partecipanti nell'ecosistema di rispettare gli standard e i requisiti di sicurezza e tecnici.

Il livello operativo primario implementa il ciclo di vita principale degli Attestati Elettronici attraverso interazioni coordinate tra Soluzioni Tecniche specializzate, in modo che tutte le

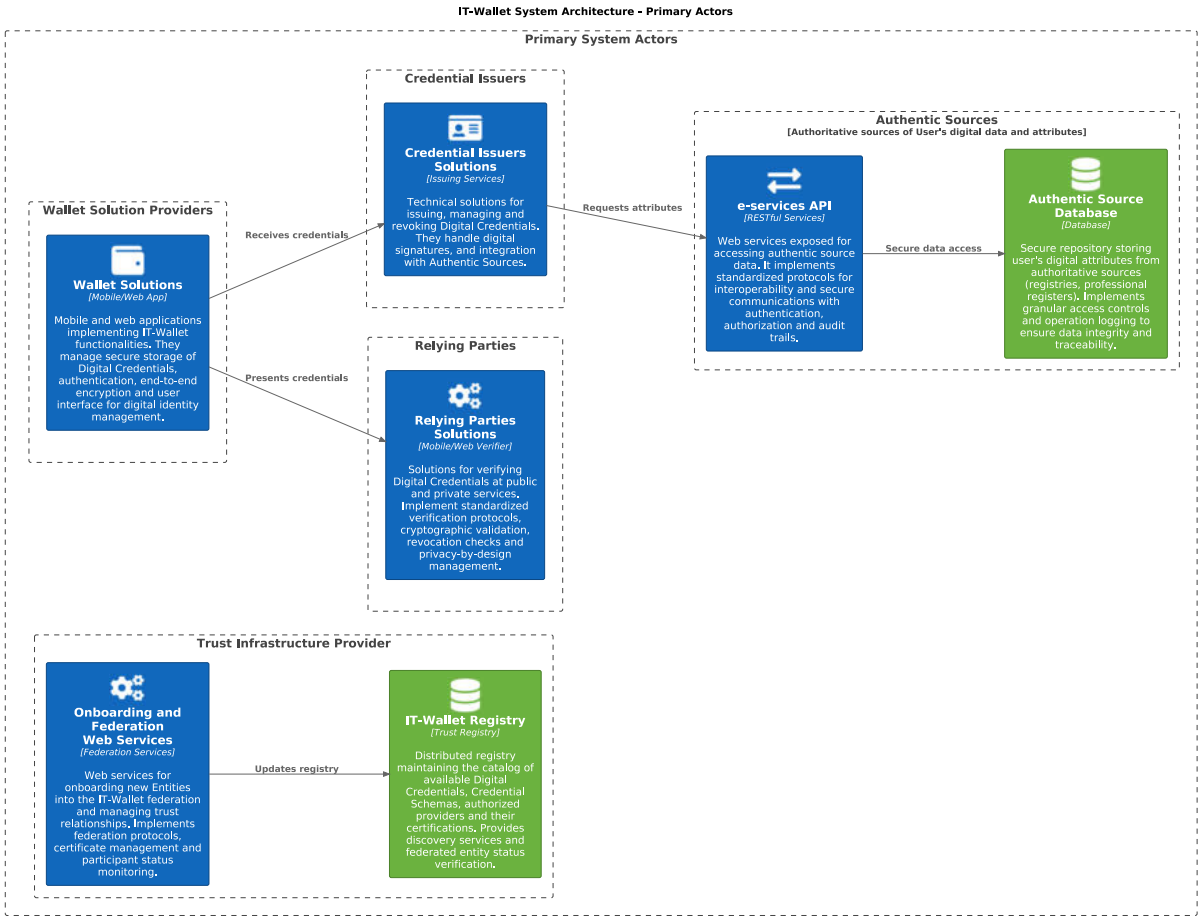


Fig. 3.2: Panoramica dell'architettura IT-Wallet - Attori Primari.

operazioni degli attestati mantengano gli standard di sicurezza e privacy consentendo al contempo esperienze utente fluide.

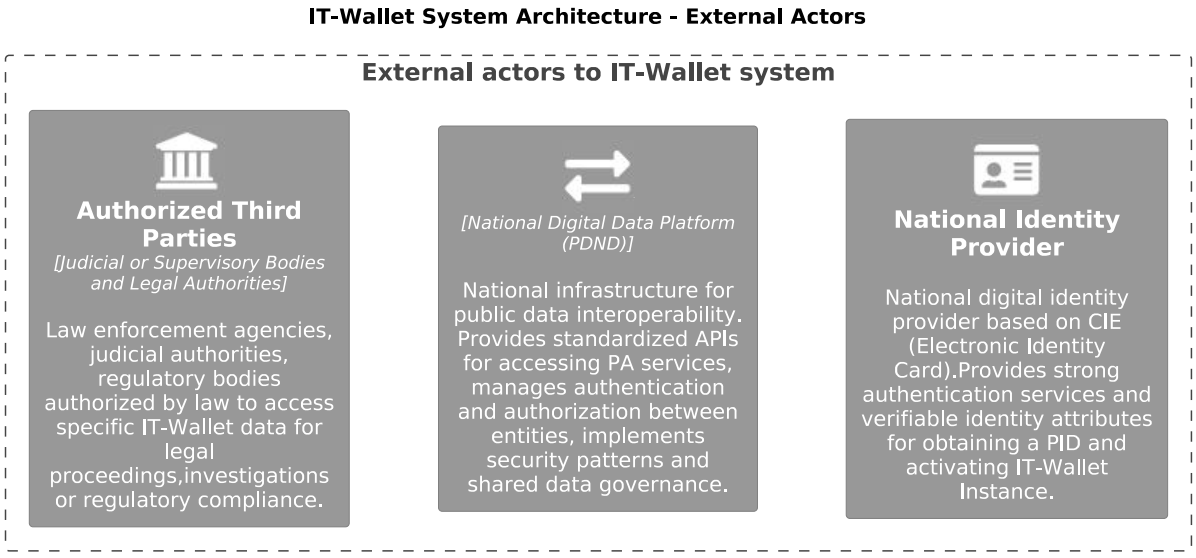


Fig. 3.3: Panoramica dell'architettura IT-Wallet - Attori esterni.

I sistemi esterni forniscono servizi che collegano l'ecosistema IT-Wallet all'infrastruttura digitale nazionale, consentendo l'interoperabilità con i servizi governativi esistenti e le fonti di dati.

Questi tre livelli architetturali lavorano insieme per consentire operazioni sicure degli Attestati Elettronici. La natura federata di questa architettura richiede processi sistematici di onboarding per stabilire relazioni di trust tra i partecipanti e standardizzare la semantica delle credenziali attraverso i componenti centralizzati Claims Registry e Taxonomy, come dettagliato in *Infrastruttura del Registro* (pagina 159).

L'architettura abilita i seguenti processi di interazione principali:

1. **Onboarding delle Entità e Federazione**: Solo le entità qualificate possono partecipare alla federazione attraverso un onboarding sistematico che garantisce la conformità e stabilisce relazioni di trust crittografiche. Il sistema di onboarding include un **Claims Registry** e una **Taxonomy** che standardizzano le definizioni semantiche per tutte le credenziali, consentendo interoperabilità e governance. L'autorizzazione delle entità segue un approccio basato su policy che concede ambiti operativi specifici basati sulle caratteristiche organizzative e sulla conformità normativa. Vedere Sezione *Sistema di Onboarding* (pagina 25) e *Infrastruttura del Registro* (pagina 159).
2. **Discovery delle Credenziali e Gestione del Catalogo**: L'ecosistema supporta meccanismi duali di discovery delle credenziali pubbliche attraverso il *Catalogo degli Attestati Elettronici* (pagina 190) per credenziali di uso generale, e discovery di Attestati Elettronici privati tramite Credential Offer dirette per casi d'uso specifici. Il catalogo viene popolato automaticamente basandosi sulle policy dell'Organismo di Supervisione dalle definizioni registrate nel Claims Registry e nella Taxonomy.

3. **Rilascio degli Attestati**: I Credential Issuer si coordinano con le Fonti Autentiche tramite API standardizzate per richiedere Attributi dell'Utente verificati, creando Attestati Elettronici basati su dati autorevoli. Il rilascio può procedere sia per attestati pubblicati nel catalogo che per offerte di attestati privati.

4. **Archiviazione e Gestione degli Attestati**: Le Soluzioni IT-Wallet ricevono e gestiscono gli Attestati Elettronici sui dispositivi degli utenti, consentendo una gestione sicura del ciclo di vita degli attestati.

5. **Presentazione e Verifica degli Attestati**: Gli Utenti presentano gli Attestati Elettronici alle Relying Party, che verificano i claim attraverso validazione crittografica e verifica dello stato. Le Relying Party operano all'interno di ambiti autorizzati che definiscono quali tipi di attestati e scopi possono richiedere, con validazione contro il *Registro dei Claims* (pagina 162) per la verifica dello schema.

CAPITOLO 4

Sistema di Onboarding

Il sistema di onboarding registra le entità operative (PID Provider, Attestation Provider, Relying Party, Fornitori di Wallet e Fonti Autentiche) affinché possano partecipare all'ecosistema IT-Wallet con autorizzazioni e relazioni di trust chiaramente definite.

4.1 Architettura del Sistema di Onboarding

Il servizio di onboarding DEVE fornire processi di onboarding specializzati che corrispondano alle caratteristiche operative e agli obblighi normativi dei diversi tipi di partecipanti.

Tutti gli Attori Primari DEVONO sottoporsi alla registrazione amministrativa per la conformità legale e normativa, seguita da processi di registrazione tecnica specializzati che DEVONO riflettere i loro ruoli operativi nell'ecosistema degli Attestati Elettronici.

1. **Registrazione Amministrativa**: Tutte le entità (Fonti Autentiche, Relying Party, Fornitori di Wallet, Credential Issuer) DEVONO completare la registrazione amministrativa iniziale che convalida la loro posizione legale, conformità normativa ed eleggibilità organizzativa per partecipare all'ecosistema IT-Wallet.
2. **Registrazione Tecnica**: Dopo l'approvazione amministrativa, le entità completano la registrazione tecnica attraverso Journey specializzati:
    a. **Fonti Autentiche**: Dichiarano i loro Attributi dell'Utente disponibili dal Registro degli Attributi dell'Utente e specificano gli scopi di verifica previsti e tipo di organizzazione (pubblica o privata) previsti.

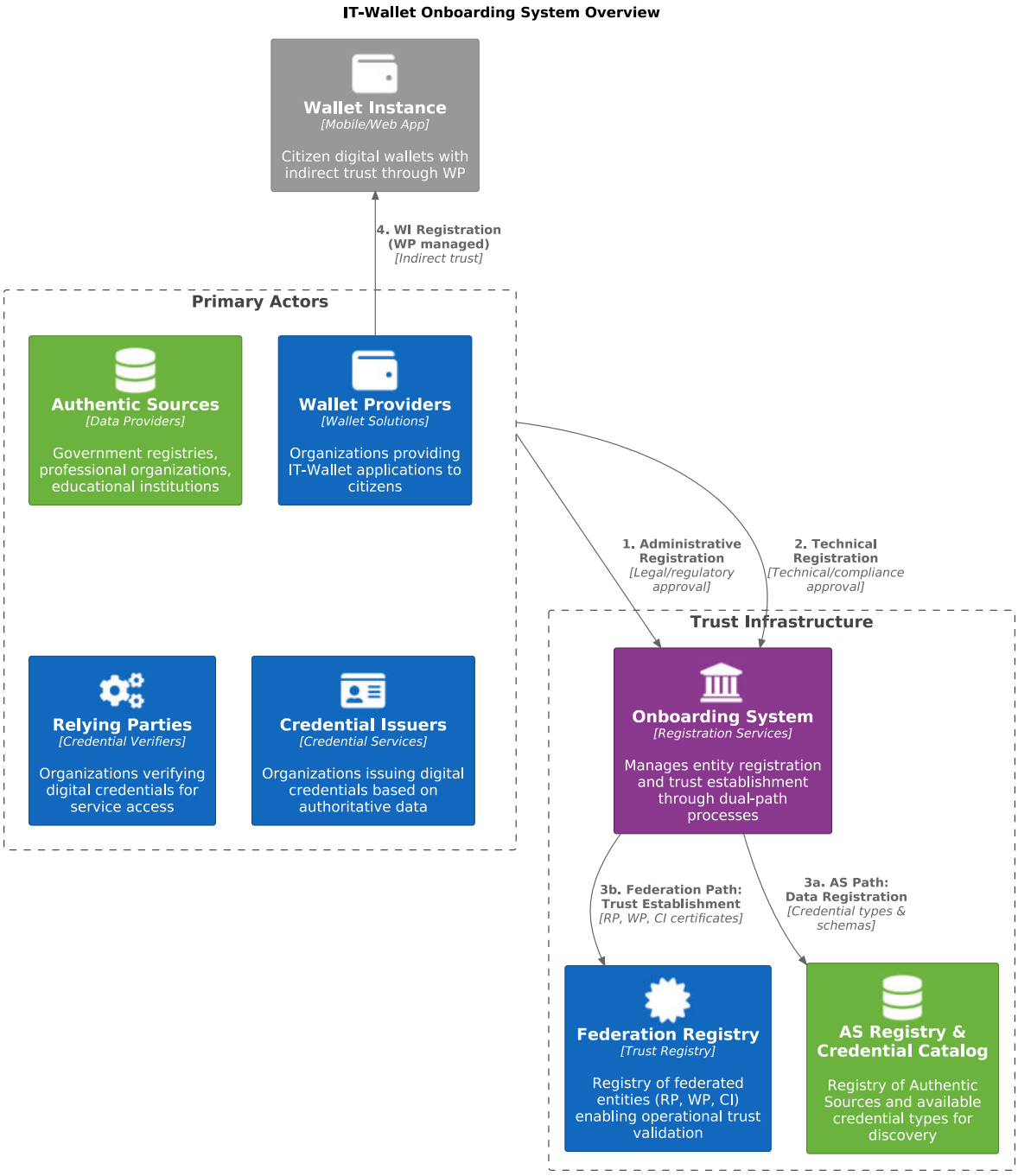


Fig. 4.1: Panoramica del Sistema di Onboarding IT-Wallet.

b. **Credential Issuer**: Selezionano le Fonti Autentiche basate sugli Attributi dell'Utente richiesti, richiedono l'approvazione dell'integrazione (eccetto per i mandati normativi) e registrano i tipi di Credenziali con pubblicazione automatica del catalogo secondo la politica dell'Organismo di Supervisione.

c. **Altre Entità di Federazione** (Relying Party, Fornitori di Wallet): Subiscono registrazione basata su federazione per l'istituzione di trust crittografico.

3. **Integrazione del Registro IT-Wallet**:

   a. **Integrazione del Registro degli Attributi dell'Utente e della Tassonomia**: Il Registro degli Attributi dell'Utente fornisce definizioni di dati standardizzate per i singoli Attributi delle Credenziali, mentre la Tassonomia definisce la classificazione gerarchica (domini, classi, scopi) che vengono poi referenziati nel Catalogo degli Attestati Elettronici per implementazioni specifiche di Credenziali. Tutti i partecipanti sfruttano questi registri per dichiarazioni di capacità di fornitura dati e requisiti di emissione/verifica.

   b. **Integrazione del Registro AS**: Fonti Autentiche registrate con i loro Attributi dell'Utente dichiarati e relative specifiche, abilitando la discovery e coordinazione dei CI.

   c. **Integrazione del Registro di Federazione**: Entità operative incluse per la validazione del trust durante le operazioni delle Credenziali.

   d. **Integrazione del Catalogo**: Tipi di Credenziali automaticamente pubblicati nel *Catalogo degli Attestati Elettronici* (pagina 190) basato sulle politiche dell'Organismo di Supervisione per l'eleggibilità alla discovery pubblica.

4. **Registrazione dell'Istanza del Wallet**: Le Istanze del Wallet sono registrate indirettamente attraverso i Fornitori di Wallet, stabilendo le specifiche di gestione delle Credenziali a livello utente. I dettagli tecnici sono forniti in *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304).

4.2 Processo di Registrazione della Fonte Autentica

La registrazione della Fonte Autentica consente ai fornitori di dati di stabilire il loro ruolo autorevole nell'ecosistema degli Attestati Elettronici attraverso la registrazione delle loro specifiche di fornire dati e meccanismi di accesso standardizzati basati sul Registro degli Attributi dell'Utente e sulle classificazioni della Tassonomia.

Le Fonti Autentiche DEVONO sottoporsi a procedure di registrazione che convalidano la loro autorità sui dati, dichiarano i loro Attributi dell'Utente disponibili dal Registro degli Attributi dell'Utente standardizzato e stabiliscono meccanismi di integrazione tecnica. Le Fonti Autentiche

specificano casi d'uso previsti (formalmente `purposes`) che determinano l'eleggibilità del catalogo secondo le politiche dell'Organismo di Supervisione.

Le Fonti Autentiche Pubbliche e Private DEVONO sfruttare l'integrazione PDND per fornire dati governativi attraverso l'infrastruttura nazionale standardizzata.

Questo percorso DEVE assicurare gli standard di qualità dei dati e stabilire audit trail per tutte le attività di fornitura dei dati.

Processo di Coordinamento AS-CI: Dopo la registrazione AS, i Credential Issuer identificano entità AS adatte attraverso il Registro AS e richiedono autorizzazione all'integrazione durante la fase di registrazione amministrativa. Per i mandati normativi, l'autorizzazione DEVE essere automatica. Altrimenti, le entità Fonti Autentiche valutano e autorizzano le richieste dei Credential Issuer basate su criteri commerciali e tecnici. Dopo l'autorizzazione amministrativa, le procedure di integrazione tecnica stabiliscono le relazioni operative di accesso ai dati prima della pubblicazione del catalogo dei tipi di Credenziali.

Le Fonti Autentiche registrate con successo DEVONO essere incluse nel Registro AS con i loro Attributi dell'Utente dichiarati e le relative specifiche. I tipi di Credenziali DEVONO diventare pubblicamente scopribili nel *Catalogo degli Attestati Elettronici* (pagina 190) solo dopo l'integrazione AS-CI riuscita e l'approvazione della politica dell'Organismo di Supervisione per l'eleggibilità del catalogo.

Le procedure di implementazione tecnica per la registrazione della Fonte Autentica sono fornite in *Procedura di registrazione delle Fonti Autentiche* (pagina 244).

4.3 Processo di Onboarding della Federazione

L'onboarding della federazione, inclusi i processi di registrazione amministrativa e tecnica, stabilisce le relazioni di trust crittografiche e le verifiche di conformità che consentono a tutte le entità (PID Provider, Attestation Provider, Relying Party, Fornitori di Wallet e Fonti Autentiche) di partecipare ad attività sicure del ciclo di vita delle Credenziali. Le entità operative DEVONO completare l'onboarding che include la verifica dell'eleggibilità amministrativa e la validazione tecnica. Per i Fornitori di Wallet, questo include la valutazione di conformità della Soluzione Wallet e i successivi flussi di notifica nelle Trusted Lists.

Per PID Provider, Attestation Provider e Relying Party, il processo di onboarding della federazione tipicamente include:

- **Registrazione dell'Entità**: Raccolta dei dati principali (identificazione, diritti, punti di fornitura del servizio e chiavi pubbliche crittografiche e certificati) necessari per autorizzare le entità e descriverne le capacità.

- **Emissione dei Certificati**: Emissione di certificati che autenticano le entità, fanno riferimento al registro per la verifica dei diritti, supportano la Certificate Transparency e descrivono lo stato di registrazione per Relying Party e Credential Issuer.
- **Pubblicazione del Registro**: Pubblicazione di tutte le entità registrate nel registro, con un'API online (vedi *Integrazione dell'Infrastruttura di Trust e del Registro* (pagina 120)) che può essere utilizzata per verificare la registrazione delle entità e gli attributi richiesti.

4.3.1 Onboarding degli Intermediari di Relying Party

Gli *Intermediari di Relying Party* seguono un processo di onboarding dedicato che conferisce lo status di Intermediario riconosciuto e l'autorità di effettuare l'onboarding delle Relying Party subordinate. Nel profilo implementativo descritto dalle presenti specifiche tecniche, un Intermediario di Relying Party corrisponde a un Intermediario di Federazione (vedi la definizione di **Intermediario IT-Wallet**). Il processo comprende:

- **Registrazione amministrativa**: L'Intermediario presenta documentazione organizzativa e legale che ne attesta l'idoneità ad agire per conto di Relying Party ai sensi dell'Art. 5b(8) del Regolamento eIDAS2 (EU_2024_1183).
- **Registrazione tecnica**: L'Intermediario registra le proprie chiavi di federazione e gli endpoint, abilitandolo a pubblicare la propria Entity Configuration e a emettere Subordinate Statement per le Foglie subordinate.
- **Emissione del Trust Mark di Intermediario**: Al termine del processo di onboarding, il Trust Anchor emette all'Intermediario un Trust Mark con identificativo `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier_intermediary`. Questo Trust Mark costituisce l'evidenza verificabile del ruolo riconosciuto dell'Intermediario nella federazione.
- **Autorizzazione all'emissione di Trust Mark**: Il Trust Anchor aggiorna la propria Entity Configuration includendo l'Intermediario nell'attributo `trust_mark_issuers` per l'identificativo `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier`, autorizzando l'Intermediario a emettere Trust Mark alle proprie Relying Party affiliate.

L'Entity Configuration della RP affiliata DEVE includere il Trust Mark emesso dall'Intermediario e DEVE impostare `authority_hints` utilizzando l'identificativo URL dell'Intermediario.

4.4 Gestione del Ciclo di Vita delle Entità

Dopo l'onboarding riuscito, le entità richiedono una gestione continua del ciclo di vita per mantenere lo stato operativo e la conformità all'interno dell'ecosistema IT-Wallet. La gestione del ciclo di vita comprende aggiornamenti amministrativi, modifiche tecniche e processi di uscita dalla federazione che soddisfano i requisiti organizzativi e operativi in evoluzione.

4.4.1 Gestione delle Operazioni Continuative

Le entità DEVONO mantenere informazioni amministrative e tecniche aggiornate per assicurare la partecipazione alla federazione e la conformità.

Aggiornamenti Amministrativi

Le organizzazioni POSSONO aggiornare le informazioni amministrative attraverso canali di registro standard, come:

- **Cambiamenti dell'Entità Legale**: Cambiamenti del nome dell'azienda, ristrutturazione organizzativa, modifiche dello stato legale.
- **Informazioni di Contatto**: Aggiornamenti ai canali di contatto ufficiali e al personale responsabile.
- **Stato Normativo**: Cambiamenti in licenze, certificazioni o stato di conformità normativa.
- **Ambito del Servizio**: Modifiche alle offerte di servizio o alle caratteristiche della base utenti.

Gli aggiornamenti amministrativi DEVONO seguire processi di governance standard e NON DOVREBBERO influenzare le operazioni tecniche della federazione.

Gestione della Configurazione Tecnica

Gli aggiornamenti tecnici che influenzano le operazioni del protocollo di federazione richiedono procedure coordinate, inclusi:

- **Gestione dei Certificati**: Rinnovo regolare dei certificati, sostituzione di emergenza, gestione della revoca.
- **Cambiamenti dell'Infrastruttura**: Aggiornamenti degli endpoint, migrazioni di servizio, modifiche della capacità.
- **Aggiornamenti di Conformità**: Aggiornamenti degli standard di sicurezza, cambiamenti delle politiche, requisiti di audit.
- **Modifiche delle Capacità**: Aggiunta o rimozione di protocolli supportati, tipi di Credenziali o caratteristiche del servizio.

Gli aggiornamenti tecnici DEVONO essere validati dall'Organismo di Supervisione per mantenere le relazioni di trust della federazione e l'integrità operativa.

4.4.2 Processi di Uscita e Rimozione dalla Federazione

Le entità POSSONO uscire dalla federazione volontariamente o essere rimosse dall'Organismo di Supervisione per ragioni di conformità o sicurezza.

Uscita Volontaria dalla Federazione - Le organizzazioni POSSONO scegliere di uscire dalla federazione per ragioni commerciali o operative, inclusi:

- **Cambiamenti Commerciali**: Ristrutturazione organizzativa o interruzione del servizio.
- **Migrazione Tecnica**: Passaggio a soluzioni tecniche o fornitori alternativi.
- **Cambiamenti Normativi**: Cambiamenti nell'ambiente normativo o nei requisiti di conformità.

L'uscita volontaria DEVE richiedere coordinamento con entità dipendenti e gestione appropriata delle Credenziali esistenti e delle relazioni con gli utenti.

Rimozione dell'Organismo di Supervisione - L'Organismo di Supervisione PUÒ avviare la rimozione dell'entità per:

- **Violazioni di Conformità**: Mancato mantenimento della conformità normativa o aderenza alle politiche della federazione.
- **Incidenti di Sicurezza**: Compromissione dell'infrastruttura di sicurezza o mancato mantenimento degli standard di sicurezza.
- **Fallimenti Operativi**: Fallimenti tecnici persistenti che influenzano la sicurezza della federazione.
- **Violazioni delle Politiche**: Violazioni delle politiche operative della federazione o degli accordi.

I processi di rimozione POSSONO includere indagini, rimedi e procedure di appello dove appropriato.

Avvertimento

Per incidenti di sicurezza critici o minacce immediate all'integrità della federazione, l'Organismo di Supervisione PUÒ implementare la sospensione di emergenza con effetto immediato.

4.4.3 Requisiti di Coordinamento del Ciclo di Vita

La gestione del ciclo di vita nell'ecosistema IT-Wallet necessita di coordinamento tra più partecipanti per mantenere le operazioni funzionanti e mantenere le relazioni di trust. Il framework di coordinamento copre tre aree principali:

- **Comunicazione degli stakeholder:** Quando le entità devono apportare cambiamenti che impattano le operazioni dell'ecosistema IT-Wallet, DEVONO informare l'Organismo di Supervisione e i partecipanti della federazione rilevanti in anticipo.
- **Sincronizzazione del registro:** Quando un'entità apporta cambiamenti che influenzano altre entità, tutti i sistemi di registro DEVONO essere aggiornati correttamente assicurando che tutti i cambiamenti siano registrati in modo sicuro con timestamp e ragioni.
- **Garanzia di continuità aziendale:** Le entità DOVREBBERO bilanciare tra l'apportare aggiornamenti necessari e mantenere i loro obblighi verso utenti e normative. Questo include assicurare che il servizio sia il più disponibile possibile, gestire i dati personali correttamente durante i cambiamenti e rimanere conformi ai requisiti legali anche in situazioni di emergenza.

Le procedure tecniche e i requisiti di conformità specifici per la gestione del ciclo di vita sono dettagliati nella Sezione *Onboarding delle Entità* (pagina 239).

4.5 Onboarding Journey Maps

Le seguenti mappe dei Journey forniscono una vista dettagliata dell'esperienza di onboarding dalla prospettiva di ogni tipo di entità e dei loro operatori. Queste rappresentazioni grafiche aiutano le organizzazioni a comprendere i passaggi specifici, i requisiti e le interazioni che incontreranno durante il loro processo di onboarding.

Ogni mappa del Journey mostra il processo completo dalla pianificazione iniziale all'integrazione finale del registro, evidenziando i punti di contatto critici con l'Organismo di Supervisione e le dipendenze tra i diversi processi di onboarding delle entità.

4.5.1 Panoramica dell'Ecosistema

Il sistema Registro IT-Wallet coordina tutte le registrazioni attraverso cinque componenti principali:

1. **Registro degli Attributi dell'Utente** per definizioni semantiche standardizzate dei singoli Attributi delle Credenziali.
2. **Registro AS** per fonti di dati e capacità di fornitura dati.
3. **Registro di Federazione** per relazioni di trust operative.

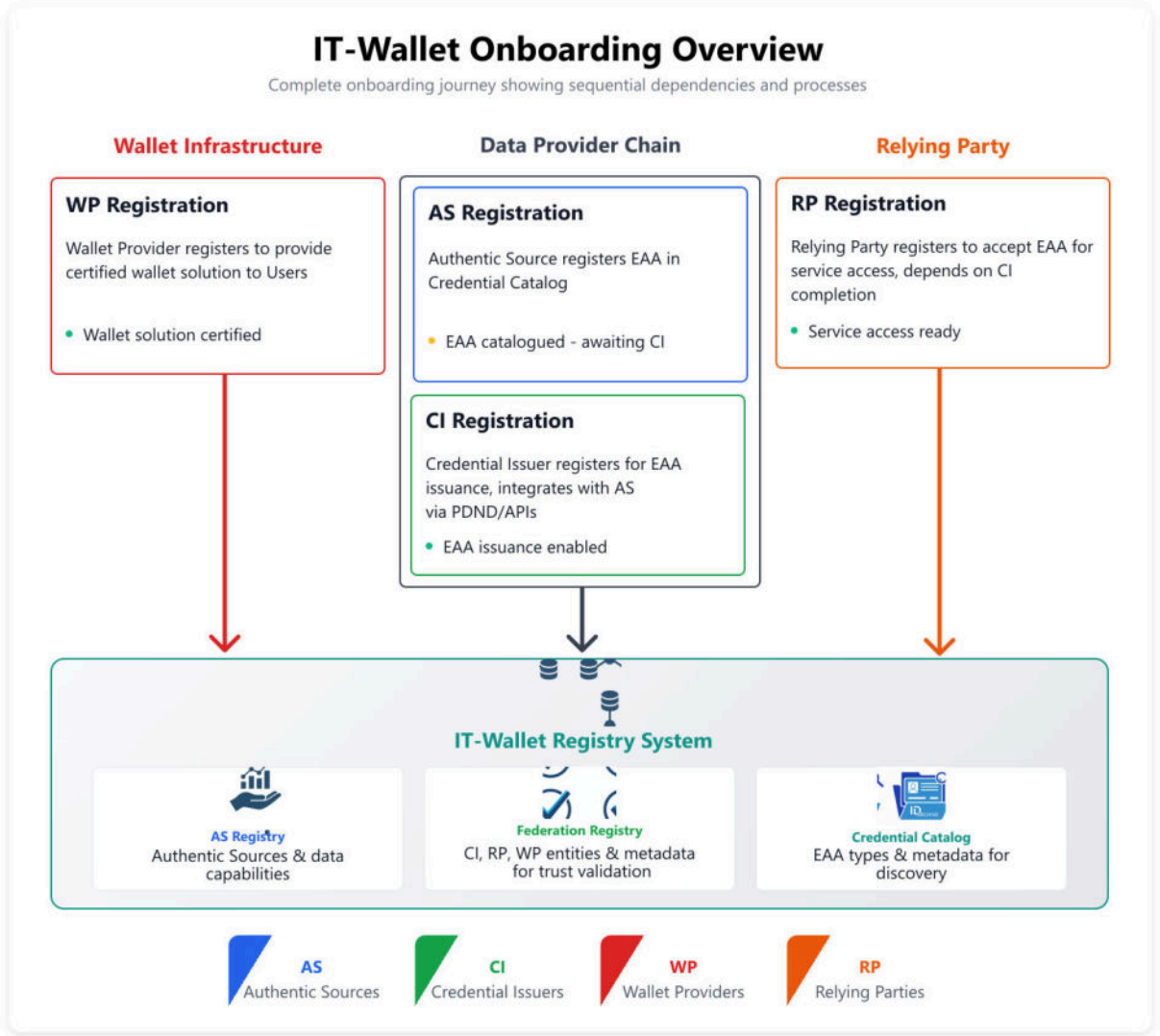


Fig. 4.2: Onboarding completo dell'ecosistema che mostra i processi principali

4. **Catalogo degli Attestati Elettronici** (vedi *Catalogo degli Attestati Elettronici* (pagina 190)) per la discovery dei tipi di Credenziali.
5. **Tassonomia** per la classificazione gerarchica che organizza le Credenziali per dominio e scopo.

Le seguenti mappe dei Journey illustrano due scenari di Credenziali distinti:

- **Scenario Catalogo Pubblico**: Patente di Guida Mobile (mDL) resa disponibile per una discovery pubblica tramite Catalogo delle Credenziali.
- **Scenario Credenziale Privata**: Badge Dipendente Aziendale dall'Azienda (FA Privata, resa disponibile per la discovery tramite Catalogo delle Credenziali ma ottenibile solo tramite Offerta di Credenziale).

4.5.2 Journey dell'Operatore della Fonte Autentica

Dalla prospettiva dell'operatore della Fonte Autentica, il processo di onboarding inizia con la valutazione delle capacità di fornire dati esistenti rispetto al Registro degli Attributi dell'Utente standardizzato e alle classificazioni della Tassonomia, determinando quali Attributi dell'Utente possono essere resi disponibili come Attestato Elettronico. L'operatore invia una richiesta di registrazione all'Organismo di Supervisione, dichiarando Attributi dell'Utente specifici dal Registro degli Attributi dell'Utente con i domini della Tassonomia e gli scopi previsti.

Esempio - Fonte Autentica Pubblica (Scenario mDL):

- **Dichiarazione degli Attributi dell'Utente**: Seleziona Attributi dell'Utente standardizzati (`given_name`, `family_name`, `driving_privileges`, ecc.) dal Registro degli Attributi dell'Utente.
- **Classificazione della Tassonomia**: Dominio `MOBILITY_TRAVEL`, Scopo `DRIVING_RIGHTS`.
- **Caso d'Uso**: Servizio pubblico - verifica dell'autorizzazione alla guida.
- **Integrazione**: Integrazione e-service PDND seguendo gli standard governativi (vedi *e-Service PDND* (pagina 714)).
- **Risultato del Catalogo**: mDL diventa pubblicamente scopribile dopo l'integrazione CI.

Esempio - AS Aziendale (Scenario Badge Dipendente):

- **Dichiarazione degli Attributi dell'Utente**: Seleziona Attributi dell'Utente (`given_name`, `family_name`, `employee.job_title`, ecc.) dal Registro degli Attributi dell'Utente.
- **Classificazione della Tassonomia**: Dominio `MEMBERSHIP`, Scopo `ASSOCIATION`.
- **Caso d'Uso**: Controllo accessi aziendale privato.
- **Integrazione**: Integrazione e-service PDND seguendo gli standard governativi (vedi *e-Service PDND* (pagina 714)).

- **Risultato del Catalogo**: Il badge diventa pubblicamente scopribile ma è reso disponibile solo tramite Offerta di Credenziale.

Le fasi critiche includono la verifica amministrativa da parte dell'Organismo di Supervisione (che coinvolge controlli di conformità normativa al di fuori dell'ambito tecnico) e la validazione tecnica. Il processo si conclude con la registrazione nel Registro AS, rendendo gli Attributi dell'Utente dichiarati scopribili dai Credential Issuer per le richieste di integrazione. Le richieste di integrazione possono essere inviate anche dalle Fonti Autentiche a specifici Credential Issuer.

Avvertimento

Dipendenza importante: Gli Attributi dell'Utente dichiarati nel Registro AS rimangono non disponibili agli utenti finali fino a quando un Credential Issuer completa la registrazione, l'approvazione dell'integrazione e l'implementazione tecnica. La pubblicazione del catalogo dipende dalle politiche dell'Organismo di Supervisione per l'eleggibilità alla discovery pubblica.

4.5.3 Journey dell'Operatore del Credential Issuer

Gli operatori del Credential Issuer iniziano effettuando la discovery delle Fonti Autentiche disponibili attraverso il Registro AS e valutando la fattibilità dell'integrazione basata sugli Attributi dell'Utente richiesti. La richiesta di registrazione specifica quali tipi di Credenziali intendono emettere, seleziona entità Fonte Autentica appropriate e dimostra la capacità tecnica di accedere alle fonti di dati richieste. In alternativa, gli operatori del Credential Issuer ricevono direttamente la richiesta di integrazione dalla Fonte Autentica.

Esempio - mDL (Scenario Pubblico):

- **discovery AS** o **AS Integration Request**: Identifica la Fonte Autentica che fornisce Attributi mDL nel Registro AS con gli Attributi dell'Utente della patente di guida richiesti. In alternativa, il Credential Issuer riceve la richiesta di integrazione direttamente dalla Fonte Authentica.
- **Richiesta di Integrazione**: Approvazione automatica dovuta al mandato normativo.
- **Configurazione Tecnica**: Integrazione e-service PDND seguendo gli standard governativi (vedi *e-Service PDND* (pagina 714)).
- **Pubblicazione del Catalogo**: mDL automaticamente pubblicato nel Catalogo delle Credenziali.
- **Accesso Utente**: I cittadini scoprono mDL attraverso un catalogo pubblico nel Wallet.

Esempio - CI per Badge Dipendente (Scenario Privato):

- **discovery AS** o **AS Integration Request**: Identifica la Fonte Autentica nel Registro AS con gli Attributi dell'Utente di accesso dipendente. In alternativa, il Credential Issuer riceve la richiesta di integrazione direttamente dalla Fonte Authentica.
- **Richiesta di Integrazione**: Richiede approvazione da parte del destinatario della richiesta.
- **Configurazione Tecnica**: Integrazione e-service PDND seguendo gli standard governativi (vedi *e-Service PDND* (pagina 714)).
- **Pubblicazione del Catalogo**: Badge automaticamente pubblicato nel Catalogo delle Credenziali.
- **Accesso Utente**: I dipendenti ricevono badge solo tramite Offerta di Credenziale diretta dai sistemi aziendali.

Dopo il test di integrazione riuscito e l'approvazione della Fonte Autentica, il Credential Issuer è registrato nel Registro di Federazione come Entità fidata, abilitando l'emissione effettiva di Credenziali agli utenti finali.

Avvertimento

Questo passaggio attiva la disponibilità delle Credenziali per gli utenti finali. Le Credenziali pubbliche e private diventano scopribili attraverso il catalogo, ma le Credenziali private saranno rese disponibili solo tramite Offerte di Credenziale.

4.5.4 Journey dell'Operatore del Fornitore di Wallet

Gli operatori del Fornitore di Wallet seguono un Journey di onboarding indipendente che si concentra sulla certificazione dell'applicazione e sulla validazione della sicurezza. Il processo evidenzia lo sviluppo e la certificazione di applicazioni Wallet che possono memorizzare e gestire in modo sicuro gli Attestati Elettronici per i cittadini.

Un requisito tecnico chiave coinvolge l'implementazione di meccanismi di controllo dell'integrità e autenticità del Wallet. Questi controlli consentono al Wallet di ottenere un Wallet Instance Attestation, che serve come prova dello stato di sicurezza e conformità del Wallet durante le operazioni delle Credenziali.

Il processo di certificazione include la valutazione della sicurezza, coprendo l'architettura del Wallet, i meccanismi di protezione dei dati e le caratteristiche di privacy dell'utente. I fornitori di Wallet certificati con successo sono registrati nel Registro di Federazione e possono distribuire le loro applicazioni attraverso gli app store.

4.5.5 Journey dell'Operatore della Relying Party

Gli operatori della Relying Party iniziano identificando quali tipi di EAA sono richiesti per i loro servizi specifici e valutando la complessità dell'integrazione con i sistemi di autenticazione esistenti. La richiesta di registrazione fornisce evidenza di necessità legittime per accedere a tipi di Credenziale specifici e Attributi dell'Utente, delineando i casi d'uso del servizio previsti e le caratteristiche organizzative che giustificano l'accesso alle Credenziali.

Esempio - Servizio di Noleggio Auto (RP Privata):

- **Classificazione Aziendale**: Codice ATECO 77.11 (servizi di noleggio auto).
- **Richiesta di Autorizzazione**: Verifica dell'autorizzazione alla guida per l'eleggibilità al noleggio.
- **Requisiti degli Attributi dell'Utente**: `given_name`, `family_name`, `driving_privileges`, ecc., da mDL.
- **Giustificazione del Caso d'Uso**: Obbligo legale di verificare la patente di guida valida prima del noleggio del veicolo.
- **Ambito di Autorizzazione**: Concesso accesso al dominio `MOBILITY_TRAVEL`, scopo `DRIVING_RIGHTS`.

Esempio - Servizi Municipali (RP Pubblica):

- **Tipo di Organizzazione**: Pubblica Amministrazione con requisito codice IPA.
- **Richiesta di Autorizzazione**: Verifica dell'identità del cittadino per l'accesso ai servizi municipali.
- **Requisiti degli Attributi dell'Utente**: `given_name`, `family_name`, `personal_administrative_number` da PID.
- **Giustificazione del Caso d'Uso**: Erogazione di servizi pubblici che richiede identificazione del cittadino.
- **Ambito di Autorizzazione**: Concesso accesso più ampio riflettendo il mandato del servizio pubblico.

Esempio - Controllo Accessi Aziendale (RP Privata):

- **Classificazione Aziendale**: Codice ATECO 62.01 (attività di programmazione informatica).
- **Richiesta di Autorizzazione**: Verifica del dipendente per l'accesso alle strutture aziendali.
- **Requisiti degli Attributi dell'Utente**: `given_name`, `family_name`, `employee.job_title` dal Badge Dipendente Aziendale.

- **Giustificazione del Caso d'Uso**: Sicurezza del posto di lavoro che richiede identificazione del dipendente e controllo degli accessi.
- **Ambito di Autorizzazione**: Concesso accesso al dominio `MEMBERSHIP`, scopo `ASSOCIATION`.
- **Discovery delle Credenziali**: Badge disponibile solo tramite Offerta di Credenziale privata (non eleggibile per il Catalogo delle Credenziali).

L'integrazione tecnica si concentra sullo sviluppo di flussi di autenticazione che possono verificare gli Attestati Elettronici presentati dagli Utenti. Questo include l'implementazione di meccanismi di verifica crittografica e l'istituzione di canali di comunicazione sicuri con l'infrastruttura di federazione.

L'autorizzazione del servizio da parte dell'Organismo di Supervisione DEVE coinvolgere una valutazione basata su politiche che considera il tipo di organizzazione (privata vs pubblica amministrazione), la classificazione del settore aziendale e i requisiti di servizio legittimi. Il processo di autorizzazione concede ambiti operativi specifici che definiscono quali domini di Credenziali e scopi la Relying Party può richiedere. Dopo l'approvazione, la Relying Party è registrata nel Registro di Federazione con profili di autorizzazione chiaramente definiti per l'accettazione degli Attestati Elettronici e degli Attributi dell'Utente.

4.6 Journey dell'Esperienza Utente

Quando tutti i processi di onboarding delle entità sono completati con successo, gli Utenti possono scoprire e installare Istanze del Wallet certificate, ottenere gli Attestati Elettronici disponibili e presentare i loro Attestati elettronici a Fornitori di Servizi registrati (vedi *Panoramica delle Funzionalità* (pagina 40)).

Questa modalità di integrazione dipende dal fatto che tutte le entità rilevanti abbiano completato i rispettivi Journey di onboarding.

CAPITOLO 5

Design dell'Esperienza Utente

5.1 Principi di Design

Il Sistema IT-Wallet si basa sul paradigma del Portafoglio di Identità Digitale e ha l'obiettivo di fornire agli Utenti un'esperienza di accesso ai servizi più semplice, veloce e sicura.

La Soluzione Wallet rappresenta il principale Touchpoint e consente agli Utenti di utilizzare e gestire i propri dati di identificazione personale e i propri Attributi nelle interazioni con soggetti pubblici o privati, sia in contesti fisici che digitali. Il corretto funzionamento del servizio, tuttavia, si fonda su un'ampia rete di interazioni tra diversi Touchpoint e Soluzioni Tecniche, che ne supportano l'erogazione e incidono direttamente sull'Esperienza Utente.

Per garantire un'elevata qualità del servizio dal punto di vista dell'Esperienza Utente, gli Attori Primari, siano essi enti pubblici o privati, DEVONO assicurare l'usabilità e l'accessibilità delle proprie Soluzioni Tecniche, in allineamento con gli elementi distintivi del Sistema IT-Wallet. Nello specifico:

- **Usabilità**: le Soluzioni Tecniche DEVONO essere progettate e mantenute per garantire elevanti standard di usabilità, allo scopo di facilitare l'adozione del servizio e ridurre la necessità di supporto. Gli enti pubblici DEVONO aderire alle [LG_DESIGN], mentre gli enti privati POSSONO prenderle a riferimento come buone pratiche;
- **Accessibilità**: le Soluzioni Tecniche DEVONO essere progettate e mantenute per garantire elevati standard di accessibilità, allo scopo di garantire accesso al servizio indipendentemente da capacità o competenze tecnologiche individuali o da vincoli esterni e di contesto. Gli enti pubblici DEVONO aderire alle [RIF_ACCESSIBILITÀ], mentre gli enti privati DEVONO rispettare quanto previsto dalla normativa vigente;

- **Coerenza**: le Soluzioni Tecniche DEVONO essere progettate e mantenute in coerenza con la Brand Identity del Sistema IT-Wallet e i requisiti funzionali relativi all'Esperienza Utente descritti in queste specifiche, per promuovere la riconoscibilità dei componenti, garantire la coerenza complessiva del sistema e minimizzare il carico cognitivo per l'Utente.

5.2 Panoramica delle Funzionalità

Il Sistema IT-Wallet offre all'utente un'esperienza più semplice, veloce e sicura di accesso ai servizi. Tale servizio si concretizza per l'Utente nella possibilità di utilizzare una Soluzione Wallet, la cui esperienza di fruizione è scandita in tre macro-fasi: pre-utilizzo, utilizzo e post-utilizzo.



Fig. 5.1: Fasi dell'Esperienza Utente di utilizzo di un'Istanza del Wallet

Le sezioni a seguire si focalizzano sulle macro-fasi di utilizzo e post-utilizzo. Esse definiscono i requisiti funzionali a supporto dell'Esperienza Utente relativi alle fasi di attivazione ottenimento, presentazione, gestione e disattivazione, unitamente ai requisiti di interazione con il servizio in termini di gestione degli errori, richiesta di assistenza e raccolta feedback.

La documentazione e le risorse aggiuntive sono disponibili nella sezione official-resources:Risorse Ufficiali.

Le Risorse Ufficiali descrivono le modalità di interazione Utente-Istanza del Wallet e le buone pratiche di progettazione al fine di promuovere coerenza tra le diverse Soluzioni Wallet, in termini di modalità di fruizione delle funzionalità.

Per garantire un'implementazione corretta e coerente, gli Attori Primari:

- DEVONO utilizzare esclusivamente le official-resources:Risorse Ufficiali e DEVONO rispettare tutte le relative specifiche di utilizzo fornite;
- POSSONO scegliere quale configurazione implementare, tra quelle rese disponibili, ma DEVONO comunque garantire il corretto utilizzo dei componenti atomici come l'*Engagement Button* (pagina 53) o l'*Authentication Button* (pagina 96);

- DEVONO garantire il costante aggiornamento delle risorse utilizzate, in linea con l'ultima versione delle official-resources:Risorse Ufficiali resa disponibile.

5.3 Attivazione dell'Istanza del Wallet

L'attivazione è il passaggio necessario per abilitare l'Utente all'utilizzo delle funzionalità della Soluzione Wallet per l'ottenimento, la presentazione e la gestione dei propri Attestati Elettronici in modo sicuro. Il processo di attivazione consiste in un'operazione di Autenticazione dell'Utente sull'Istanza del Wallet tramite la propria identità digitale (vedere *Ciclo di vita dell'Istanza del Wallet* (pagina 298)). Dopo l'Autenticazione, l'Utente ottiene un Attestato di identificazione personale — il **PID** EUDI oppure l'**IT-Wallet ID** nazionale — come descritto in *Focus sugli Attestati di Identificazione Personale* (pagina 42).

Di seguito i requisiti di Esperienza Utente che il Wallet Provider DEVE garantire attraverso la propria Soluzione Wallet:

- l'Utente scarica la Soluzione Wallet sul suo dispositivo così da generare la propria Istanza del Wallet;
- l'Utente imposta un metodo di sblocco per la sua Istanza del Wallet se non è già stato impostato in precedenza nell'app. In aggiunta al PIN, l'Utente può decidere di usare il metodo di sblocco usato per il dispositivo e gestito a livello di sistema operativo (e.g. autenticazione biometrica) come alternativa al PIN. L'Utente utilizza il metodo di sblocco ogni qual volta è richiesta un'autorizzazione a garanzia della sicurezza e della protezione delle proprie informazioni;
- l'Utente prende visione di tutte le informazioni rilevanti sull'attivazione e sulle modalità di utilizzo del servizio. L'Utente inoltre prende visione di eventuali informative del Fornitore di Wallet e del fornitore dell'Attestato di identificazione personale (PID Provider o fornitore dell'IT-Wallet ID) e/o termini e condizioni d'uso del servizio. L'Utente dà il proprio consenso per proseguire oppure lo nega per annullare l'operazione;
- l'Utente sceglie una tra le modalità di Autenticazione disponibili;
- l'Utente conclude il flusso di Autenticazione sul servizio del Gestore di Identità Digitale;
- l'Utente riceve conferma dell'esito del processo di Autenticazione e, se positivo, visualizza l'anteprima dell'Attestato di identificazione personale, (PID o IT-Wallet ID). L'Utente conferma le informazioni mostrate in anteprima per procedere oppure annulla l'operazione;
- l'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- l'Utente visualizza l'esito positivo dell'avvenuta attivazione dell'Istanza del Wallet.

Il Fornitore di Wallet DEVE permettere all'Utente di rimuovere un Attestato di identificazione personale ottenuto in fase di attivazione. Inoltre, il relativo fornitore (PID Provider o fornitore dell'IT-Wallet ID) DOVREBBE permettere all'Utente di revocare tale Attestato tramite uno specifico Touchpoint. Il Fornitore di Wallet DEVE permettere all'Utente di richiedere la disattivazione della propria Istanza del Wallet, anche in assenza del dispositivo su cui è stata installata. Per approfondimenti si rimanda alle sezioni *Disattivazione dell'Istanza del Wallet* (pagina 107) e *Gestione degli Attestati Elettronici* (pagina 100).

In caso di errori nell'utilizzo della Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

5.3.1 Focus sugli Attestati di Identificazione Personale

Questa sezione definisce i requisiti di Esperienza Utente per gli Attestati di identificazione personale ottenuti nell'Istanza del Wallet: il **PID** EUDI e l'**IT-Wallet ID** nazionale (Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale). Entrambi veicolano un set minimo verificato di informazioni sull'identità dell'Utente (vedere *Modello di Dati degli Attestati Elettronici* (pagina 379)), ma differiscono per ambito giuridico, tipologia di fornitore, impatto sul ciclo di vita dell'Istanza del Wallet e di casi d'uso di presentazione supportati.

Di seguito sono riportati i requisiti di Esperienza Utente per assicurare una visualizzazione e un utilizzo dell'Attestato di identificazione personale (PID e/o IT-Wallet ID) uniforme e coerente. Il Fornitore di Wallet:

- DEVE mostrare correttamente l'Attestato di identificazione personale su tutti i dispositivi, garantendo un'esperienza coerente su schermi di dimensioni diverse;
- DEVE mostrare lo stato dell'Attestato, se diverso da valido, per fornire trasparenza sul suo ciclo di vita, e PUÒ visualizzarlo se valido. Dettagli specifici sullo stato non valido POSSONO essere forniti (ad esempio, il motivo della revoca dell'Attestato);
- DEVE includere Action Button per consentire la gestione del ciclo di vita dell'Attestato e permettere all'Utente di revocarlo o aggiornarlo in qualsiasi momento (vedere *Gestione degli Attestati Elettronici* (pagina 100));
- DEVE garantire che l'Attestato sia un elemento dispositivo, affinché l'Utente lo possa usare per autenticarsi presso una Relying Party in un contesto digitale (vedere *Autenticazione* (pagina 80)), per accedere ai servizi in contesti di prossimità ove applicabile e per richiedere l'emissione di ulteriori EAA ove il relativo fornitore richieda dati di identificazione personale (vedere *Ottenimento degli Attestati Elettronici di Attributi* (pagina 46));
- DEVE mostrare un metodo di assistenza reso disponibile dal relativo fornitore dell'Attestato (vedere *Assistenza Utente* (pagina 112));

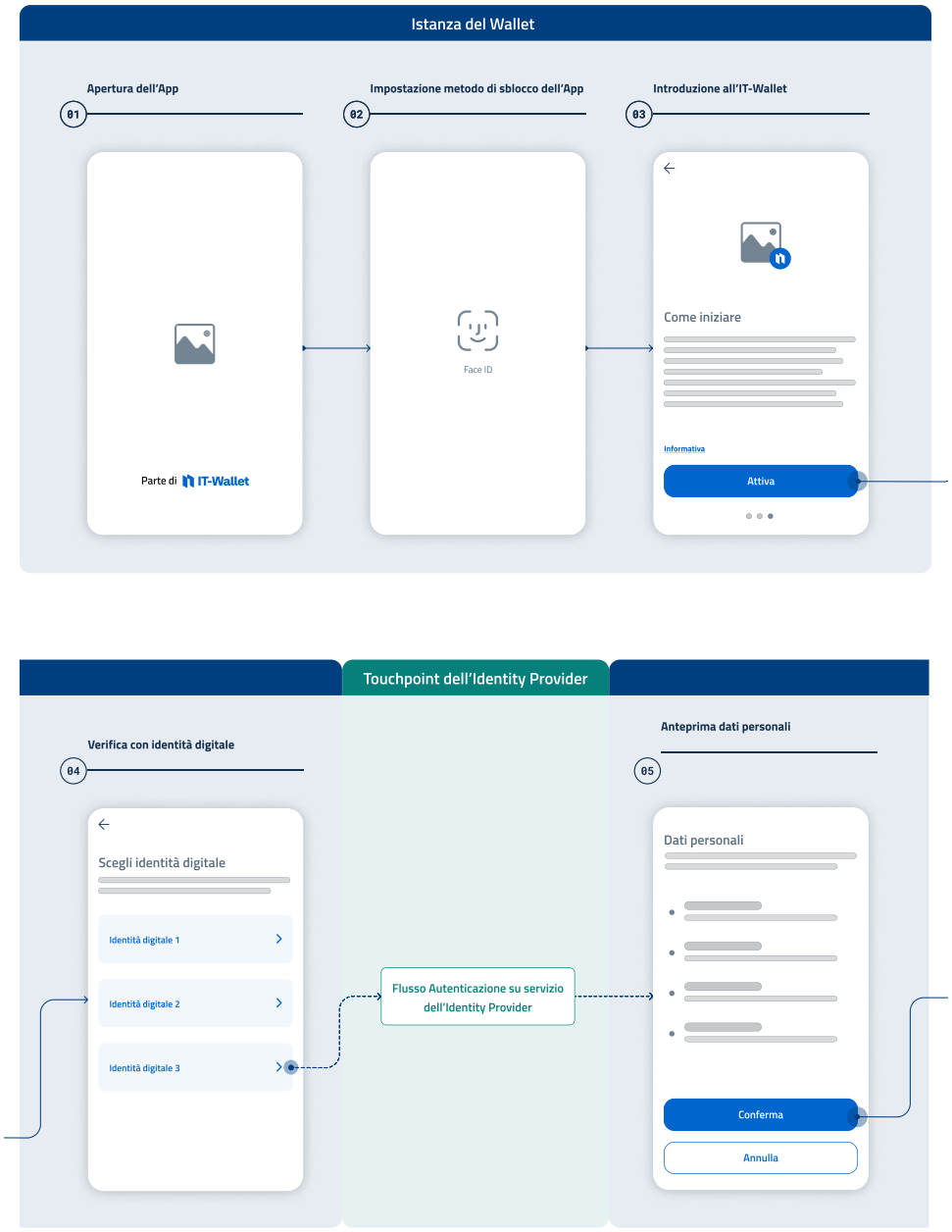


Fig. 5.2: Esempio di Esperienza Utente nell'Attivazione di un'Istanza del Wallet - 01

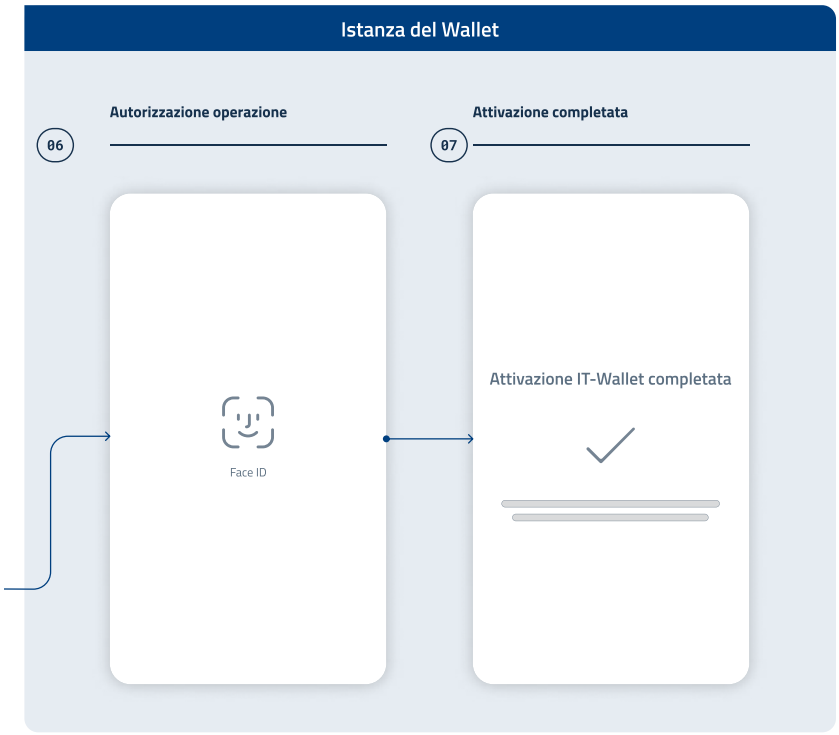


Fig. 5.3: Esempio di Esperienza Utente nell'Attivazione di un'Istanza del Wallet - 02

- DEVE garantire che l'Attestato sia riconoscibile dall'Utente e distinguibile da altri EAA.

PID

Il **PID** è l'insieme dei Dati di Identificazione Personale nel quadro dell'Identità Digitale Europea (vedere *Modello di Dati del PID* (pagina 404) e *Dati di Identificazione Personale*). È rilasciato dal PID Provider come PID tecnico (struttura dati firmata) per la presentazione dall'Istanza del Wallet.

- L'Istanza del Wallet in stato **Operativo** consente l'emissione del PID. A seguito del rilascio con esito positivo di un PID valido, l'Istanza del Wallet DEVE transitare nello stato **Valido** (vedere *Transizione a Valido* (pagina 302)).
- Finché l'Istanza del Wallet è **Valida** e il PID è attivo/valido, l'Utente PUÒ presentare il PID e usarlo per l'autenticazione e per ulteriori emissioni di Attestati secondo la presente specifica e il quadro EUDI.
- La revoca, la scadenza o la cancellazione del PID DEVONO far transitare l'Istanza del Wallet di nuovo nello stato **Operativo** (vedere *Transizione a Operativo* (pagina 301)).
- Qualora l'Utente revochi il PID, il Fornitore di Wallet DEVE gestire le conseguenze sul ciclo di vita dell'Istanza del Wallet e sulle EAA dipendenti come definito nella presente

specifica.

- Il PID è destinato anche a interazioni cross-border nell'ecosistema EUDI Wallet, oltre che a usi nazionali.

IT-Wallet ID

L'**IT-Wallet ID** è l'Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale rilasciato esclusivamente per usi nazionali (vedere *Modello di Dati dell'IT-Wallet ID* (pagina 416) e *IT-Wallet ID*). È un EAA rilasciato da un Fornitore di Attestati Elettronici di Attributi, a differenza del PID.

- L'emissione dell'IT-Wallet ID **non** fa transitare l'Istanza del Wallet nello stato **Valido**; l'Istanza del Wallet rimane **Operativa** (vedere *Transizione a Operativo* (pagina 301)).
- L'IT-Wallet ID è destinato esclusivamente ad interazioni nazionali e non cross-border.
- Per assicurare un'identificazione e una rappresentazione coerente dell'IT-Wallet ID tra tutte le Soluzioni Wallet, il Fornitore di Wallet:
  - DEVE utilizzare la denominazione ufficiale "IT-Wallet ID" nelle sue Soluzioni Wallet;
  - DEVE utilizzare l'elemento grafico ufficiale di IT-Wallet ID disponibile nelle official-resources:Risorse Ufficiali e DEVE rispettare tutte le relative specifiche di utilizzo fornite;
  - DEVE utilizzare l'elemento grafico di IT-Wallet ID in formato `application/svg+xml` data;
  - NON DEVE alterare, distorcere, modificare o sostituire l'elemento grafico di IT-Wallet ID con elementi grafici non ufficiali;
  - DEVE garantire l'area di rispetto minima definita nelle official-resources:Risorse Ufficiali, al fine di garantirne un'adeguata visibilità e riconoscibilità. Altri elementi grafici o testuali NON DEVONO interferire con questa area di rispetto;
  - NON DEVE ridimensionare l'elemento grafico di IT-Wallet ID oltre i limiti minimi stabiliti dalle official-resources:Risorse Ufficiali, in modo da garantire sempre una leggibilità ottimale su qualsiasi formato o dispositivo;
  - NON DEVE utilizzare l'elemento grafico di IT-Wallet ID su sfondi di colore che ne compromettano la visibilità o la leggibilità. DEVE garantire un contrasto adeguato tra elemento grafico di IT-Wallet ID e lo sfondo, in conformità con quanto definito nelle official-resources:Risorse Ufficiali.

  Sebbene il termine IT-Wallet ID rappresenti l'Attestato Elettronico di Dati di Identificazione personale, la risorsa grafica IT-Wallet ID PUÒ essere utilizzata dal

Fornitore di Wallet per facilitare l'Utente nell'individuazione, più in generale, dei suoi Attestati di identificazione personale (IT-Wallet ID / PID).



Fig. 5.4: Elemento grafico "IT-Wallet ID" su sfondo chiaro

La Risorsa Ufficiale dell'"IT-Wallet ID" è disponibile all'interno della Sezione official-resources: Risorse Ufficiali. Ulteriore documentazione sull'elemento grafico dell'IT-Wallet ID è disponibile nel Brand Manual, indicato nella sezione official-resources:Risorse Ufficiali.

5.4 Ottenimento degli Attestati Elettronici di Attributi

Una volta che l'attivazione è conclusa, l'Utente PUÒ richiedere e ottenere nella propria Istanza del Wallet uno o più Attestati Elettronici di Attributi.

L'Attestato Elettronico di Attributi è un oggetto dinamico che consente all'Utente di dimostrare o attestare, in modo affidabile e verificabile, una condizione, uno stato o un diritto, sulla base delle informazioni in esso contenute. Nello specifico, l'Attestato Elettronico:

- **viene definito da una Fonte Autentica**, il soggetto titolare dei dati sorgente;
- **è costituito da un insieme di Attributi**, i dati granulari che lo definiscono (no immagini statiche o pdf) quali, ad esempio, una caratteristica (es. "residente") o una qualifica (es. "studente");
- **è emesso da un Fornitore di Attestati Elettronici di Attributi**, il soggetto che lo rilascia sulla base delle informazioni acquisite e ne gestisce il ciclo di vita.

A seconda delle specifiche esigenze dell'Utente, della tipologia di Attestato Elettronico di Attributi e delle disponibilità offerte dal Fornitore di Wallet, dal Fornitore di Attestati Elettronici di Attributi e dalla Fonte Autentica, l'ottenimento degli Attestati Elettronici di Attributi può avvenire attraverso due modalità:

- **dal Catalogo dell'Istanza del Wallet**: l'Utente esplora l'elenco degli Attestati Elettronici di Attributi forniti dalla Soluzione Wallet, seleziona quello di interesse e avvia il processo di richiesta, concludendo con il rilascio dell'Attestato Elettronico di Attributi nell'Istanza del Wallet. Questo percorso è disponibile per i tipi di credenziale idonei per la

discovery pubblica come determinato dalle politiche dell'organismo di supervisione durante il processo di onboarding (vedere *Catalogo degli Attestati Elettronici* (pagina 190)).

- **da un Touchpoint della Fonte Autentica** (o del Fornitore di Attestati Elettronici di Attributi se coincide con la Fonte Autentica; vedere *Flusso Credential Offer* (pagina 477)): l'Utente interagisce con il servizio digitale della Fonte Autentica, consentendogli di ottenere un Attestato Elettronico di Attributi specifico nella propria Istanza del Wallet tramite un *Engagement Button* (pagina 53).

Nonostante le modalità di avvio della richiesta siano diverse, i flussi di ottenimento condividono una struttura e un processo simili.

5.4.1 Ottenimento dal Catalogo dell'Istanza del Wallet

Di seguito, sono illustrati i requisiti dell'Esperienza Utente del flusso di ottenimento di un Attestato Elettronico di Attributi dal Catalogo che il Fornitore di Wallet DEVE garantire attraverso la propria Soluzione Wallet:

- l'Utente accede alla propria Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata;
- l'Utente seleziona l'Attestato Elettronico di Attributi che intende aggiungere alla sua Istanza del Wallet scegliendo tra quelli disponibili nel Catalogo;
- l'Utente seleziona da quale Fornitore di Attestati Elettronici vuole ottenere l'Attestato Elettronico di Attributi, se presente più di uno;
- l'Utente visualizza eventuali informazioni aggiuntive sui requisiti e/o limitazioni relative all'ottenimento dell'Attestato Elettronico di Attributi, provenienti dalla Fonte Autentica;
- l'Utente prende visione dei suoi dati di identificazione personale, qualora richiesti dalla Fonte Autentica per l'ottenimento dell'Attestato Elettronico di Attributi, il nome del relativo Fornitore di Attestati Elettronici di Attributi e di eventuali informative. L'Utente dà il proprio consenso per poter proseguire presentando i dati richiesti o altri Attributi al Fornitore di Attestati Elettronici di Attributi oppure annulla l'operazione;
- l'Utente visualizza l'anteprima dell'Attestato Elettronico di Attributi. L'Utente conferma i dati mostrati in anteprima per procedere all'ottenimento oppure annulla l'operazione;
- l'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- l'Utente visualizza l'esito positivo dell'ottenimento avvenuto;
- l'Utente visualizza i dettagli dell'Attestato Elettronico di Attributi ottenuto ovvero: i dati in esso contenuti, il nome del Fornitore di Attestati Elettronici di Attributi che ha emesso l'Attestato Elettronico di Attributi e il nome della Fonte Autentica;

- l'Utente ha evidenza di tutti gli Attestati Elettronici ottenuti navigando la sua Istanza del Wallet.

La Fonte Autentica PUÒ fornire all'Utente delle informazioni aggiuntive relative a un Attestato Elettronico di Attributi. Tali informazioni DEVONO essere mostrate dal Fornitore di Wallet all'Utente nell'Istanza del Wallet prima di avviare l'effettivo flusso di ottenimento dell'Attestato Elettronico di Attributi. Al fine di redigere correttamente questo contenuto informativo, la Fonte Autentica:

- DEVE utilizzare un linguaggio chiaro (e.g. evitare termini tecnici o complessi), essenziale (es. evitare testi eccessivamente lunghi o articolati) e inclusivo (es. evitare i verbi abilisti) seguendo le buone pratiche di scrittura, linguaggio e tono di voce descritte nelle [RIF_ACCESSIBILITÀ] e, nel caso di enti pubblici, nelle [LG_DESIGN];
- DEVE attenersi allo scopo specifico del testo, ossia quello di comunicare informazioni utili all'Utente prima di intraprendere il processo di ottenimento (es. elencare i prerequisiti o dichiarare delle limitazioni che potrebbero influenzare il buon esito della procedura di ottenimento);
- DEVE garantire l'aggiornamento costante delle informazioni;
- DEVE prevedere un titolo e un testo all'interno del quale PUÒ includere il riferimento a canali esterni per indirizzare verso una procedura, approfondire un determinato argomento e/o aprire richieste di supporto.

Segue un esempio di testo informativo:

Nota

Titolo: Chi può ottenere il documento. **Testo:** La versione digitale del [Nome documento] è disponibile solo per coloro che possiedono già la versione fisica. Se vuoi avere maggiori dettagli, [leggi più informazioni] (URL).

Per approfondimenti si rimanda alla sezione *Registro delle Fonti Autentiche* (pagina 173) (vedi claim `data_capabilities.user_information`).

Il Fornitore di Wallet DEVE permettere all'Utente di rimuovere un Attestato Elettronico di Attributi dalla sua Istanza del Wallet in ogni momento. In caso di assenza del dispositivo su cui è stata attivata l'Istanza del Wallet, il Fornitore di Wallet DEVE permettere all'Utente di disattivare l'intera Istanza del Wallet tramite un Touchpoint dedicato. Inoltre, i Fornitori di Attestati Elettronici di Attributi DOVREBBERO permettere all'Utente la revoca degli Attestati Elettronici ottenuti, tramite specifici Touchpoint. Per approfondimenti si rimanda alle sezioni *Disattivazione dell'Istanza del Wallet* (pagina 107) e *Gestione degli Attestati Elettronici* (pagina 100).

In caso di problemi di comunicazione tra i sistemi del Fornitore di Attestati Elettronici di Attributi e della Fonte Autentica o in presenza di processi amministrativi o tecnici che non consentono di fornire immediatamente l'Attestato Elettronico di Attributi, gli attori coinvolti POSSONO gestire il flusso di ottenimento in modalità differita. In questo caso, il Fornitore di Wallet DEVE garantire che:

- l'Utente, giunto all'ultimo step del processo, visualizzi un messaggio che lo invita ad attendere il momento in cui l'Attestato Elettronico di Attributi potrà essere rilasciato;
- l'Utente venga informato dal Fornitore di Attestati Elettronici di Attributi non appena l'Attestato Elettronico di Attributi è disponibile.

In caso di dati errati all'interno di un Attestato Elettronico di Attributi già ottenuto o in fase di ottenimento, il Fornitore di Wallet DOVREBBE garantire all'Utente un'adeguata assistenza attraverso la propria Istanza del Wallet. Per approfondimenti si rimanda alla sezione *Assistenza Utente* (pagina 112).

In caso di errori nell'utilizzo della Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

5.4.2 Ottenimento dal Touchpoint della Fonte Autentica

Di seguito sono illustrati i requisiti dell'Esperienza Utente del flusso di ottenimento di un Attestato Elettronico di Attributi dal Touchpoint della Fonte Autentica (chiamato anche *Flusso Credential Offer* (pagina 477)) che questa DEVE garantire attraverso il proprio Touchpoint:

- L'Utente interagisce con l'*Engagement Button* (pagina 53) chiaramente esposto nell'interfaccia del Touchpoint;
- L'Utente seleziona la Soluzione Wallet con la quale procedere, attraverso un'interfaccia che DEVE seguire le indicazioni e le funzionalità previste per la Selection Page descritta nella sezione *Autenticazione* (pagina 80);
- (*solo cross-device*) L'Utente scansiona un QR code che invoca l'apertura dell'Istanza del Wallet prescelta, attraverso un'interfaccia che DEVE seguire le indicazioni e le funzionalità previste per la QR Code Page descritta nella sezione *Autenticazione* (pagina 80); in alternativa l'Utente può uscire da flusso.
- L'Utente accede alla propria Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente prende visione dei propri dati di identificazione personale, qualora richiesti dalla Fonte Autentica per l'ottenimento dell'Attestato Elettronico di Attributi, il nome del relativo Fornitore di Attestati Elettronici di Attributi e di eventuali informative. L'Utente

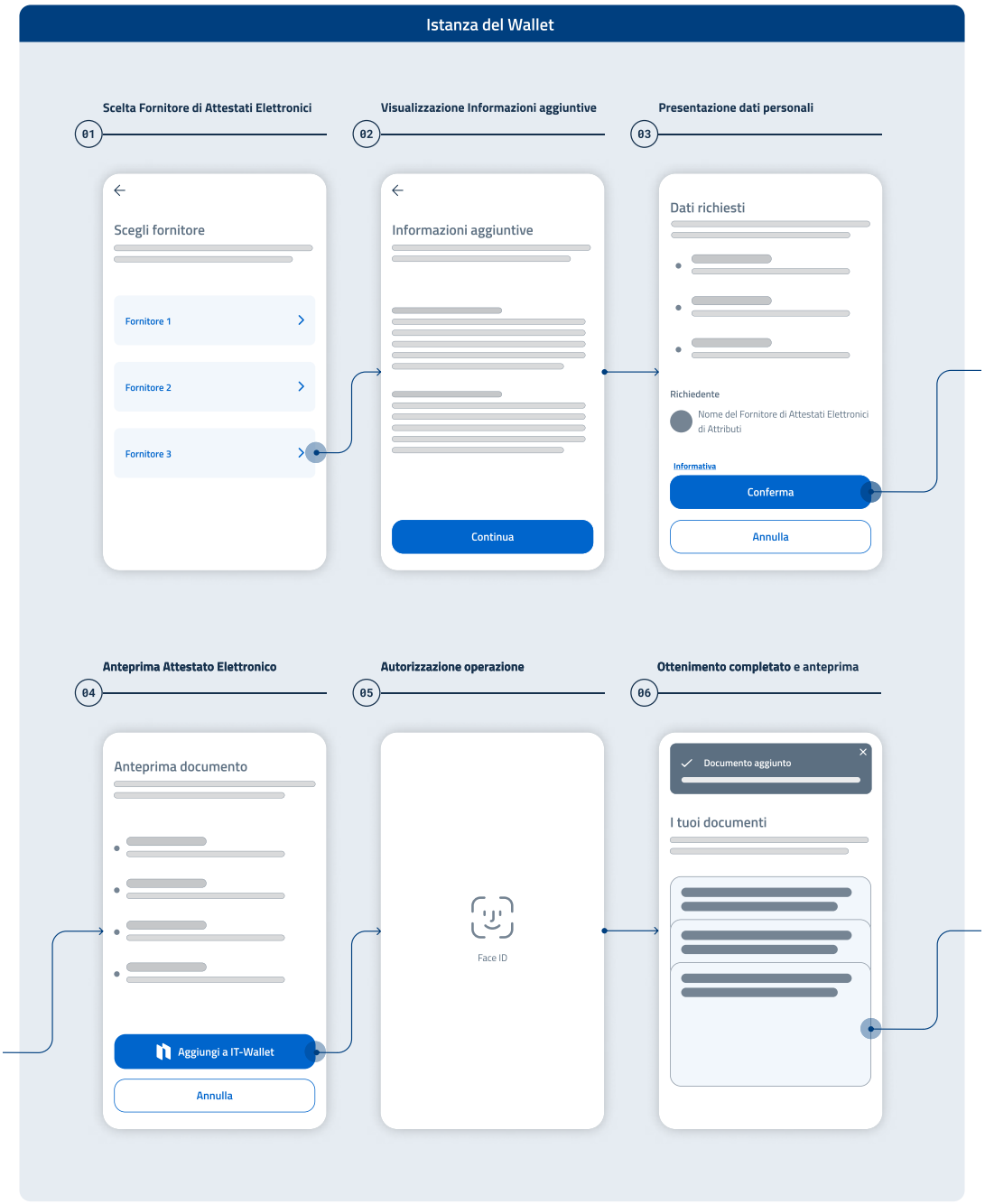


Fig. 5.5: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico di Attributi da Catalogo - 01



Fig. 5.6: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico di Attributi da Catalogo - 02

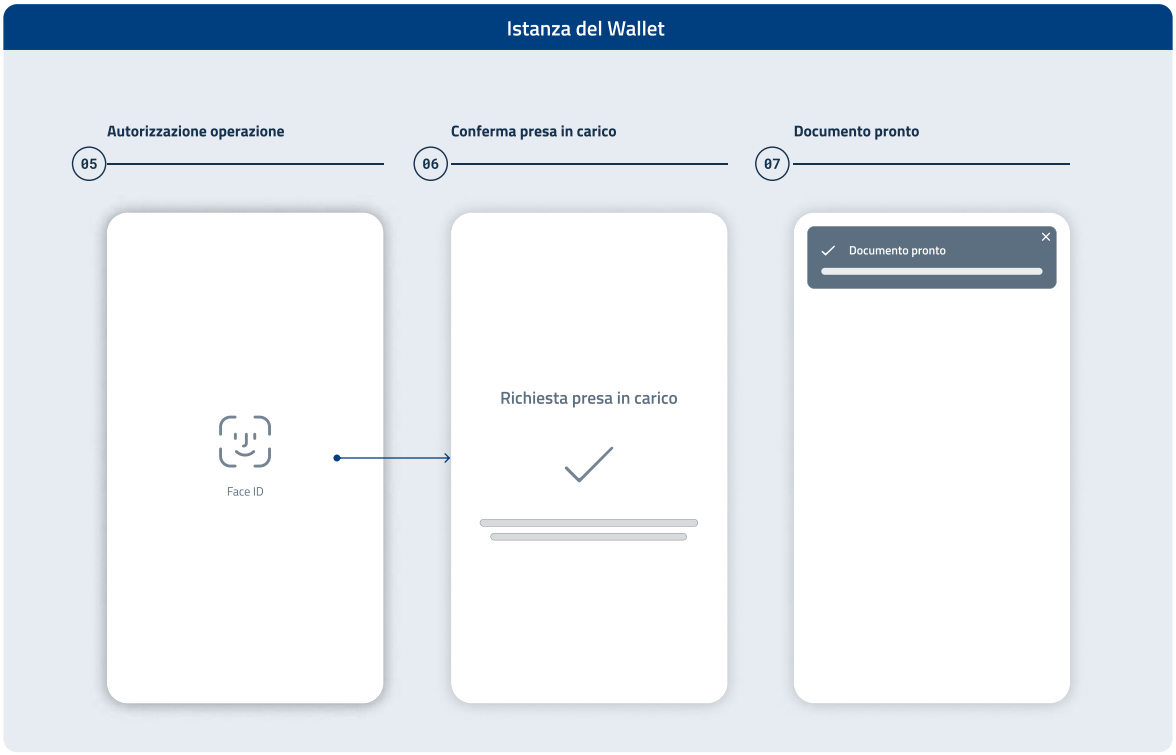


Fig. 5.7: Esempio di Esperienza Utente nell'Ottenimento differito di un Attestato Elettronico di Attributi da Catalogo - 03

dà il proprio consenso per poter proseguire presentando i dati richiesti o altri Attributi al Fornitore di Attestati Elettronici di Attributi oppure annulla l'operazione;

- L'Utente visualizza eventuali informazioni aggiuntive sui requisiti e/o limitazioni relative all'ottenimento dell'Attestato Elettronico di Attributi;
- L'Utente visualizza l'anteprima dell'Attestato Elettronico di Attributi. L'Utente conferma i dati mostrati in anteprima per procedere all'ottenimento oppure annulla l'operazione;
- L'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente visualizza l'esito positivo dell'ottenimento avvenuto;
- L'Utente visualizza i dettagli dell'Attestato Elettronico di Attributi ottenuto ovvero: i dati in esso contenuti, il nome del Fornitore di Attestati Elettronici di Attributi che ha emesso l'Attestato Elettronico di Attributi e il nome della Fonte Autentica.

In caso di errori nell'ottenimento dell'Attestato Elettronico di Attributi, la Fonte Autentica DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

Di seguito è rappresentata a titolo esemplificativo la pagina con l'*Engagement Button* (pagina 53), insieme alla Selection Page e alla QR Code Page con gli elementi dell'interfaccia e i testi adattati al flusso di *Ottenimento dal Touchpoint della Fonte Autentica* (pagina 49).

Engagement Button

L'Engagement Button è un elemento interattivo dell'interfaccia che consente all'Utente di avviare un processo, ad esempio di Autenticazione ("Entra con IT-Wallet"), di ottenimento di un Attestato Elettronico ("Aggiungi a IT-Wallet") o di verifica ("Verifica con IT-Wallet").

I Verificatori di Attestati Elettronici POSSONO rendere disponibile l'Engagement Button all'interno della pagina di un Touchpoint per permettere all'Utente di fruire dei servizi tramite un'Istanza del Wallet.

L'Engagement Button:

- DEVE essere utilizzato esattamente come indicato nelle Risorse Ufficiali e NON DEVE essere riprogettato ad hoc;
- DEVE essere utilizzato esclusivamente nelle forme, colori e proporzioni stabilite e NON DEVE essere alterato, distorto o nascosto;
- DEVE adattarsi a tutte le risoluzioni di schermo e DEVE essere integrato nella Discovery Page in modo da garantire i requisiti minimi di usabilità e accessibilità;

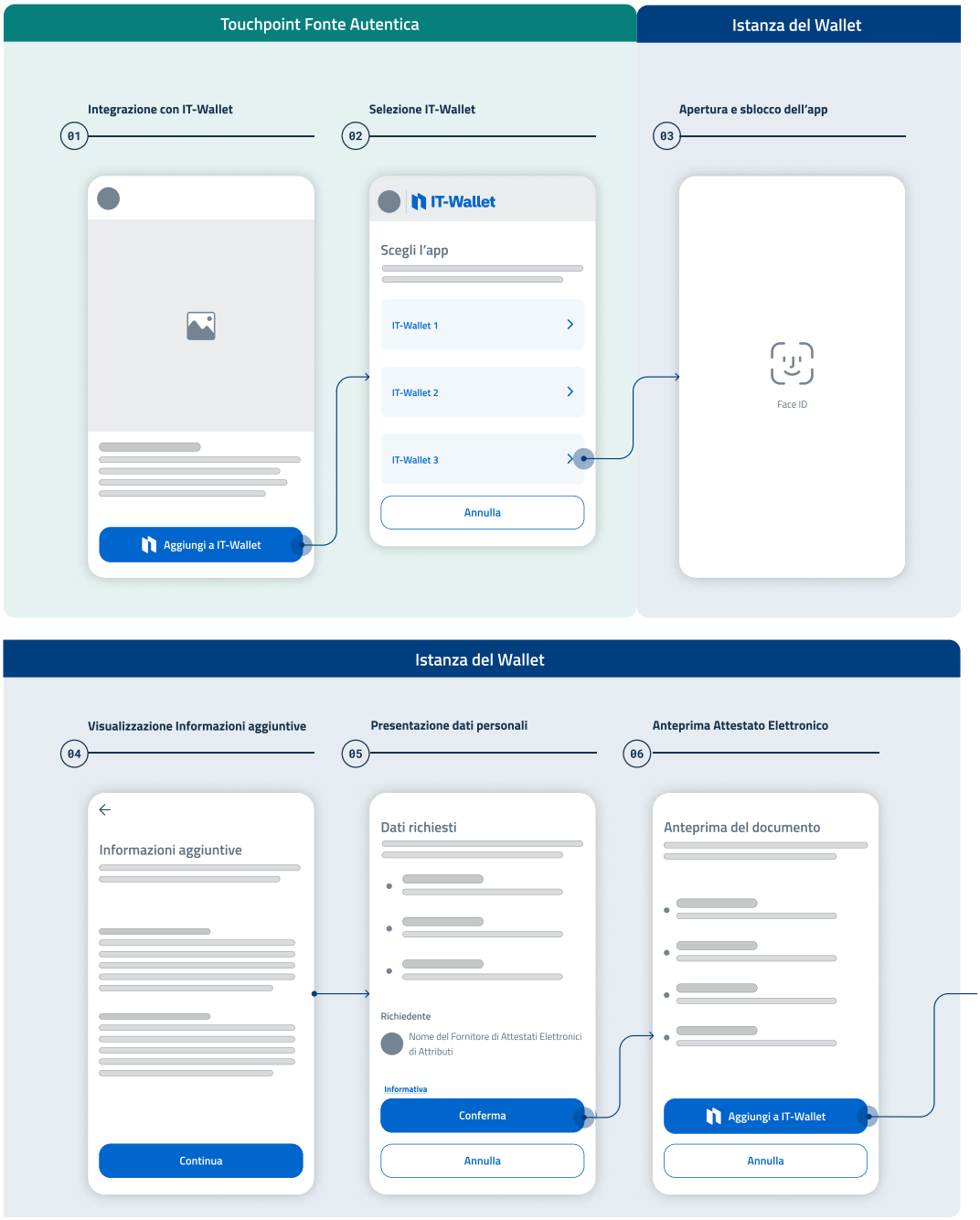


Fig. 5.8: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, same device - 01

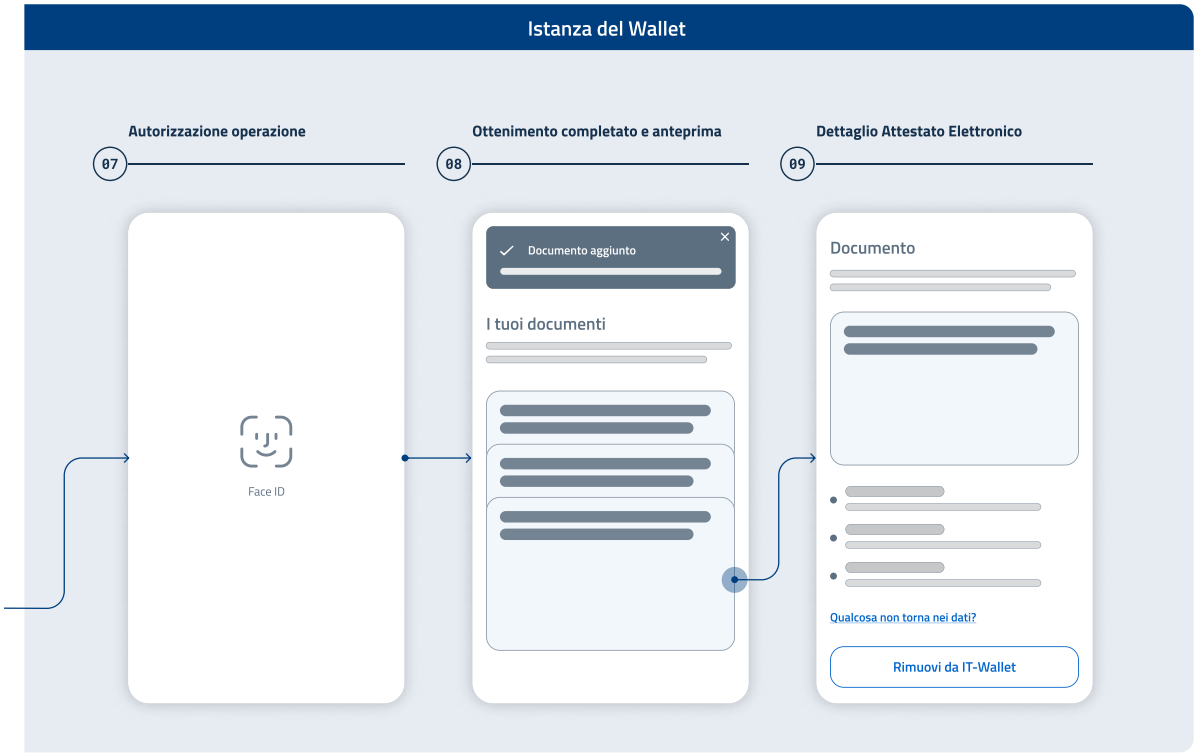


Fig. 5.9: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, same device - 02

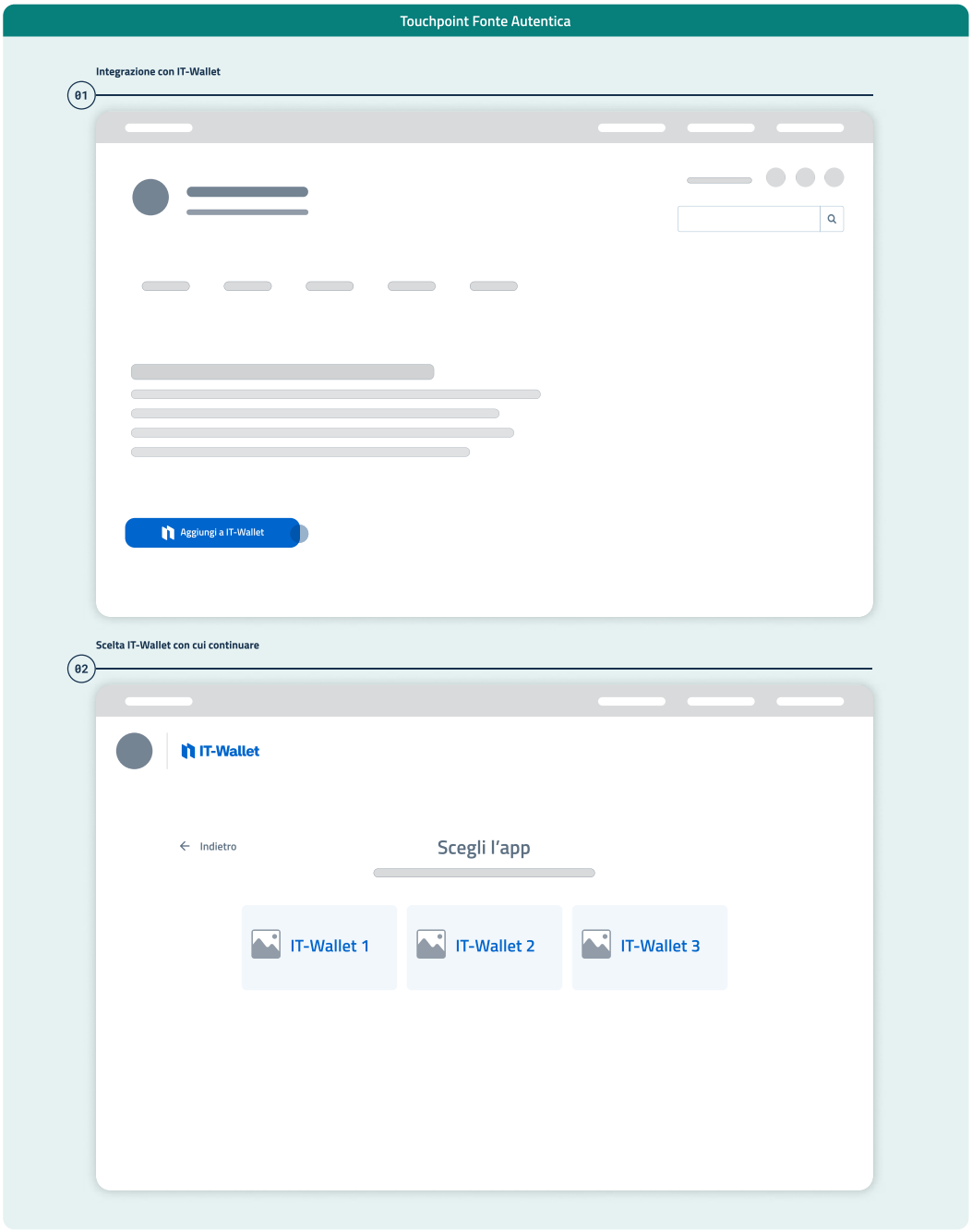


Fig. 5.10: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, cross device - 01

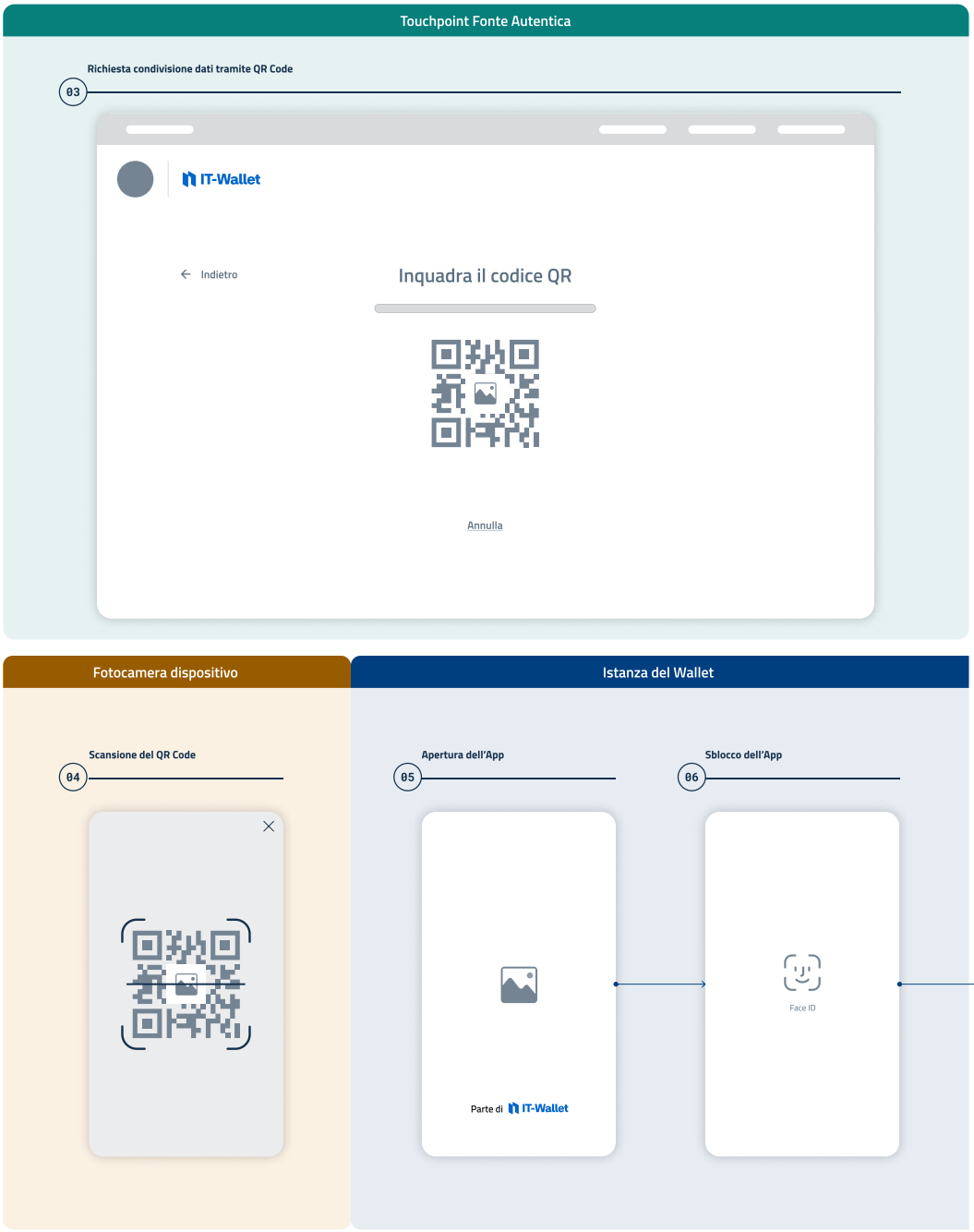


Fig. 5.11: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, cross device - 02



Fig. 5.12: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, cross device - 03

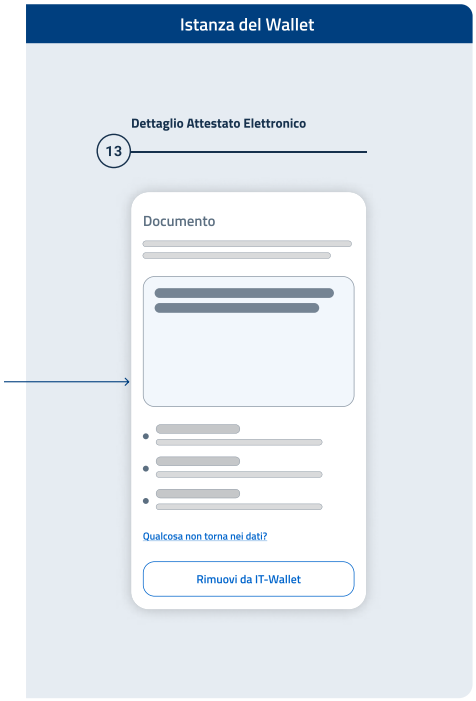


Fig. 5.13: Esempio di Esperienza Utente nell'Ottenimento di un Attestato Elettronico da Touchpoint della Fonte Autentica, cross device - 04

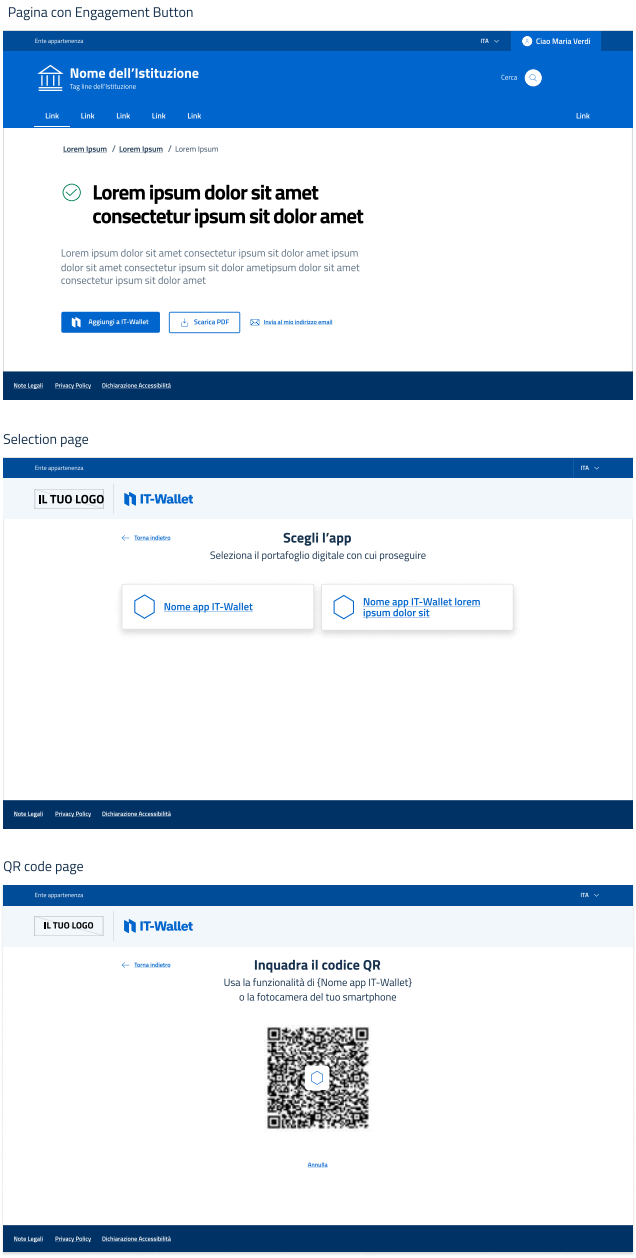


Fig. 5.14: Esempio di pagina con Engagement Button, Selection page e QR code page per il credential offer

- DEVE mantenere una distanza minima da altri elementi (quiet zone) di almeno 24px;
- DEVE riportare nella dicitura l'azione da svolgere seguita da "IT-Wallet", ad esempio "Entra con IT-Wallet", "Aggiungi a IT-Wallet", "Verifica con IT-Wallet");
- PUÒ essere accompagnato da un testo che spiega il valore aggiunto dal fare quell'azione usando il sistema IT-Wallet.
- Gli attori che intendono integrare l'Engagement Button nel proprio Touchpoint DEVONO garantirne la traduzione in altre lingue, almeno quella inglese;

Di seguito alcuni esempi non normativi di dicitura dell'Engagement Button: - Entra con IT-Wallet / Login with IT-Wallet (vedere *Authentication Button* (pagina 96))

- Aggiungi a IT-Wallet / Add to IT-Wallet (vedere *Ottenimento dal Touchpoint della Fonte Autentica* (pagina 49))
- Verifica con IT-Wallet / Verify with IT-wallet (vedere *Presentazione da remoto* (pagina 70))

5.4.3 Focus sugli Attestati Elettronici di Attributi

Progettare e rappresentare in modo adeguato un Attestato Elettronico di Attributi è fondamentale per apportare reale valore nell'intero sistema. In questo, il ruolo della Fonte Autentica e del Fornitore di Wallet è determinante.

- La Fonte Autentica è responsabile della progettazione di tutte le caratteristiche alla base dell'Attestato Elettronico di Attributi in termini di:
- **opportunità e valore percepito**: cosa l'EAA è in grado di attestare, in quali contesti d'uso e da parte di chi;
- **facilità di comprensione**: quanto è semplice e intuitivo per chi lo utilizza e per chi lo verifica;
- **efficacia comunicativa**: quanto è considerato riconoscibile o percepito come affidabile da parte di chi lo utilizza o verifica.
- Il Fornitore di Wallet è responsabile della rappresentazione dell'Attestato Elettronico di Attributi in linea con:
- quanto definito dalle presenti Specifiche Tecniche in ottica di coerenza di sistema e di aderenza ai principi di usabilità [LG_DESIGN] e di accessibilità [RIF_ACCESSIBILITÀ];
- quanto veicolato dal Registro delle Fonti Autentiche e dal Catalogo delle Credenziali Digitali (vedi *Infrastruttura del Registro* (pagina 159));
- quanto progettato a livello di UX/UI nella propria Soluzione Wallet.

Al fine di guidare la Fonte Autentica nel progettare e il Fornitore di Wallet nel rappresentare adeguatamente un EAA, si definiscono a seguire gli elementi da considerare e la struttura da adottare a livello di Vista di Dettaglio:

- **Nome EAA**, ossia la denominazione ufficiale, distintiva e comprensibile dall'Utente di un EAA;
- **Attributi identificativi** (se previsti), ossia le informazioni anagrafiche di base dell'Utente secondo l'ordine:
  - Nome
  - Cognome
  - Codice Fiscale
  - Data di nascita
  - Luogo di nascita
  - Etc.
- **Attributi I livello**, ossia i dati che rappresentano e identificano la natura specifica dell'EAA (es. Numero patente, Categoria, Data scadenza, etc.);
- **Attributi II livello** (se previsti), ossia i dati di dettaglio utili a descrivere o approfondire un determinato Attributo di I livello. Tali Attributi di II livello possono essere strutturati attraverso una:
  - **Lista di descrizioni**, ossia una lista di valori afferenti allo stesso oggetto;
  - **Lista di Attributi con valore**, ossia una lista di ulteriori oggetti con valore associato;
- **Metadati**, ossia i dati che identificano gli estremi dell'EAA a livello di:
  - **Nome Fonte Autentica**, ossia il nome identificativo del soggetto titolare o della base dati ufficiale che detiene le informazioni alla base dell'EAA, definito dal parametro `data_capabilities.data_origin`;
  - **Nome Fornitore di Attestati Elettronici di Attributi**, ossia il nome identificativo del soggetto che realizza ed emette operativamente l'EAA, definito dal parametro `organization_name`.

Di seguito sono riportati i requisiti di dettaglio per la progettazione e la rappresentazione degli EAA all'interno delle Soluzioni Wallet (vedi *Registro delle Fonti Autentiche* (pagina 173) e *Catalogo degli Attestati Elettronici* (pagina 190)). In particolare:

La Fonte Autentica:

- DEVE indicare il **nome dell'EAA** tramite il parametro `credential_name`. In particolare, la Fonte Autentica:

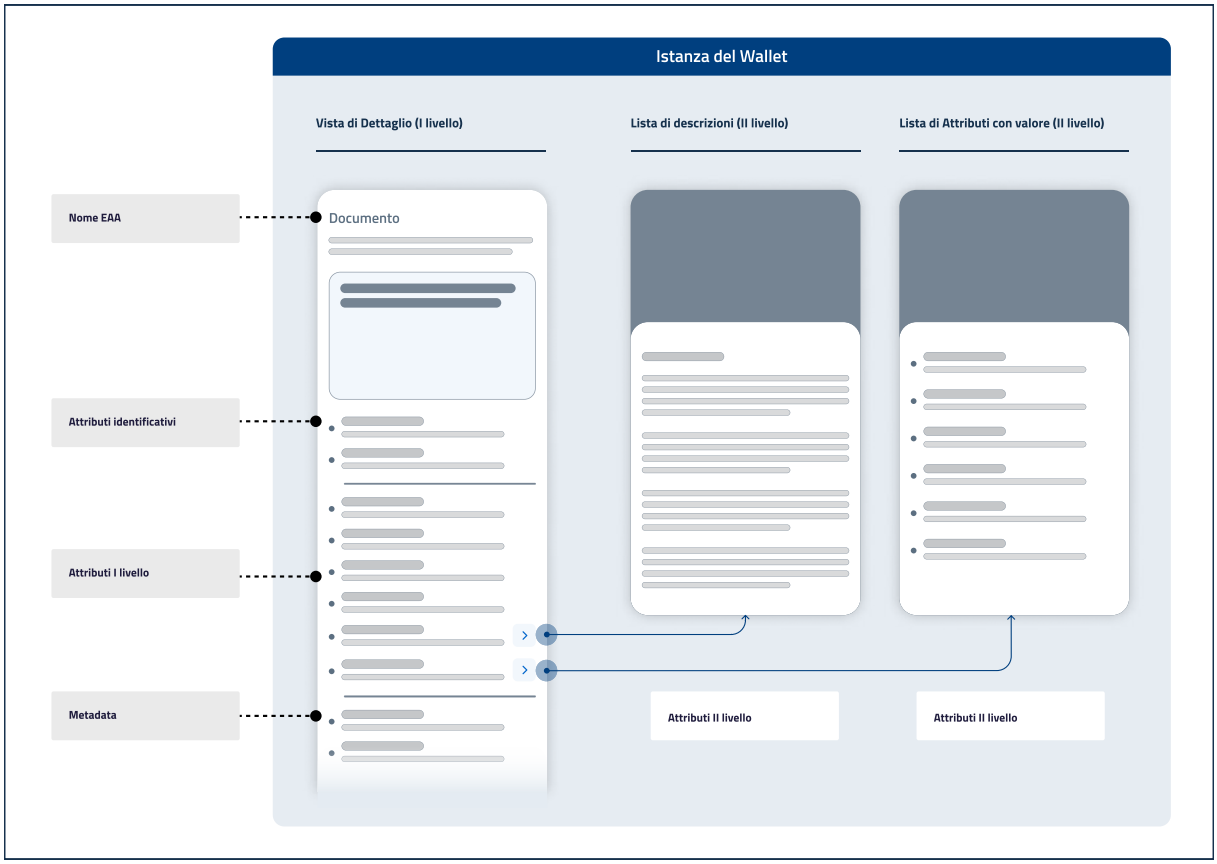


Fig. 5.15: Esempio di layout di Attestato Elettronico di Attributi, Vista di Dettaglio

  - DEVE utilizzare termini o locuzioni corrispondenti o affini al nome di una eventuale versione preesistente del documento a cui corrisponde l'EAA, così da agevolare la riconoscibilità da parte dell'Utente;
  - DEVE utilizzare termini chiari e semplici evitando, dove possibile, inglesismi, acronimi o termini tecnici;
  - DOVREBBE definire un nome breve e conciso.
- DEVE indicare il **nome del responsabile dei dati o della fonte dei dati** tramite il parametro `data_capabilities.data_origin`. In particolare, la Fonte Autentica:
  - DEVE utilizzare il nome ufficiale per garantire massima riconoscibilità e fiducia da parte dell'Utente;
  - DEVE seguire la struttura "[Acronimo] - [Nome]" in caso voglia utilizzare un acronimo unitamente al nome per esteso (es. "ANIS - Anagrafe Nazionale dell'istruzione superiore");
  - PUÒ seguire la struttura "[Acronimo soggetto primario] - [Nome soggetto secondario]" in caso sia utile disaccoppiare l'acronimo dal nome per esteso così da aggiungere un ulteriore livello di dettaglio e favorire la riconoscibilità e il senso di affidabilità del soggetto titolare (es. "MIT - Direzione Generale per la Motorizzazione").
- PUÒ rendere disponibile il **logo della Fonte Autentica** in due versioni, una versione compatta tramite il parametro `organization_info.logo_uri` e una versione estesa tramite il parametro `organization_info.logo_extended_uri`. In particolare, la Fonte Autentica:
  - DEVE fornire il logo in uno dei seguenti formati: `image/png`, `image/svg+xml` o `image/webp`;
  - DEVE fornire il logo sia in versione positiva che negativa, se disponibile;
  - DEVE fornire il logo con una dimensione minima di 30 × 30 pixel e una dimensione massima di 60 x 60 pixel, nella sua versione compatta;
  - DEVE fornire il logo secondo una ratio di 1:1, nella sua versione compatta;
  - DEVE fornire un logo che non ecceda il peso massimo di 80 KB, nella sua versione compatta;
  - DEVE fornire il logo con una dimensione minima di 200 × 30 pixel e una dimensione massima di 650 × 180 pixel, nella sua versione estesa;
  - DEVE fornire un logo che non ecceda il peso massimo di 150 KB, nella sua versione estesa.
- PUÒ rendere disponibile un **logo dell'EAA** tramite il parametro `data_capabilities.logo_uri`. In particolare, la Fonte Autentica:

  - DEVE fornire il logo in uno dei seguenti formati: `image/png`, `image/svg+xml`, or `image/webp`;
  - DEVE fornire il logo sia in versione positiva che negativa, se disponibile;
  - DEVE fornire il logo con una dimensione minima di 200 × 30 pixel e una dimensione massima di 650 x 180 pixel;
  - DEVE fornire un logo che non ecceda il peso massimo di 150 KB.
- PUÒ definire un **colore dell'EAA** tramite il parametro `data_capabilities.background_color`. In particolare, la Fonte Autentica:
  - DEVE fornire il colore utilizzando esclusivamente una delle seguenti modalità colore: HEX, HSB, RGB, sRGB, HSL, o HSV.
- DEVE fornire l'**ordine degli Attributi** desiderato tramite il parametro `data_capabilities.available_claims_order`. L'obiettivo è quello di caratterizzare l'EAA rendendolo riconoscibile agli occhi dell'Utente, soprattutto nei casi in cui si voglia dare a un nuovo EAA una forte immagine identitaria oppure si voglia mantenere coerenza con un eventuale corrispettivo fisico (es. si pensi alla patente di guida italiana e al suo caratteristico colore rosa). In particolare, la Fonte Autentica:
  - DEVE ordinare i dati secondo la struttura predefinita:
    * Attributi identificativi (opzionali);
    * Attributi I livello;
    * Attributi II livello (opzionali).
  - NON DEVE superare i due livelli di annidamento nell'organizzazione degli Attributi;
  - NON DOVREBBE superare il numero massimo di 15 Attributi per ciascun livello;
  - NON DOVREBBE superare un massimo di 85 caratteri per la valorizzazione degli Attributi;
  - NON DOVREBBE superare un massimo di 181 caratteri per la valorizzazione di ciascuna descrizione in caso di Attributi di II livello con "lista di descrizioni".

Il Fornitore di Wallet:

- DEVE mostrare in modo chiaro e accessibile il **nome dell'EAA** così come definito dal parametro `credential_name` all'interno del *Catalogo degli Attestati Elettronici* (pagina 190);
- DEVE mostrare in modo chiaro e accessibile tutti gli **Attributi dell'EAA** rispettando la macro-struttura predefinita a livello di Vista di Dettaglio, la suddivisione degli Attributi tra I e II livello, se prevista, e l'ordinamento degli Attributi come definito dal parametro `data_capabilities.available_claims_order` all'interno del *Registro delle Fonti Autentiche* (pagina 173). In particolare, la Fornitore di Wallet:

  - DEVE rappresentare l'EAA secondo la struttura predefinita:
    * Nome EAA
    * Attributi identificativi (opzionali)
    * Attributi I livello
    * Attributi II livello (opzionali)
    * Metadati
  - DEVE includere il **logo della Fonte Autentica** e/o il **logo dell'EAA** specifico se forniti dalla Fonte Autentica per mezzo del *Registro delle Fonti Autentiche* (pagina 173);
  - DEVE garantire l'adozione del colore dell'EAA definito dalla Fonte Autentica se indicato all'interno del *Registro delle Fonti Autentiche* (pagina 173) a livello di tonalità colore (H). Il Wallet Provider PUÒ ottimizzare i valori di saturazione (S) e luminosità (B) per adattare il colore specifico ad esigenze legate all'accessibilità e/o alle scelte grafiche di prodotto della propria Soluzione Wallet. Qualora la Fonte Autentica non fornisca specifiche cromatiche, il Wallet Provider è tenuto a definire e adottare proprie scelte grafiche di default.

Il Fornitore di Wallet è il responsabile ultimo della resa grafica degli Attestati Elettronici di Attributi nelle proprie Soluzioni Wallet. Pertanto, per garantire un elevato livello di accessibilità [RIF_ACCESSIBILITÀ] e usabilità [LG_DESIGN], di seguito sono riportati ulteriori requisiti di Esperienza Utente. In particolare, il Fornitore di Wallet:

- DEVE mostrare l'EAA correttamente su tutti i dispositivi, garantendo un'esperienza coerente su schermi di dimensioni diverse;
- PUÒ mostrare gli EAA ottenuti in formato tessera nella Vista in Anteprima, in linea con gli approcci già utilizzati da altri portafogli digitali nel mercato;
- DEVE garantire un layout dell'EAA ottimizzato per scalabilità e usabilità, nella Vista in Anteprima, specialmente quando più EAA vengono visualizzati sulla stessa schermata;
- DEVE mostrare chiaramente il nome dell'EAA, così come definito dal Catalogo delle Credenziali Digitali attraverso il parametro *credential_name* (vedi *Catalogo degli Attestati Elettronici* (pagina 190)), sia nella Vista di Anteprima che nelle Vista di Dettaglio;
- DEVE mostrare lo stato dell'EAA, se diverso da valido, per fornire trasparenza sul suo ciclo di vita, e PUÒ visualizzarlo se valido, sia nella Vista di Anteprima che nella Vista di Dettaglio. Evidenze specifiche sullo stato dell'EAA, se non valido, POSSONO essere fornite nella Vista di Dettaglio (ad esempio, il motivo per cui l'EAA è stato revocato);
- PUÒ includere informazioni opzionali per migliorare l'Esperienza Utente e la riconoscibilità dell'EAA, sia nella Vista di Anteprima che nella Vista di Dettaglio, come ad esempio il

logo e/o il colore, come definito della Fonte Autentica attraverso il Registro delle Fonti Autentiche (vedi *Registro delle Fonti Autentiche* (pagina 173));

- DEVE includere le stesse informazioni della Vista di Anteprima nella Vista di Dettaglio e dare una rappresentazione esaustiva di tutti gli altri Attributi, seguendo l'ordine definito dalla Fonte Autentica nel Registro delle Fonti Autentiche (vedi *Registro delle Fonti Autentiche* (pagina 173)); PUÒ includere elementi informativi aggiuntivi nella Vista di Dettaglio, ad esempio informazioni relative agli scenari di utilizzo o al motivo dell'eventuale invalidità dell'EAA;
- DEVE includere Action Button nella Vista di Dettaglio per consentire la gestione del ciclo di vita dell'EAA e permettere all'Utente di revocare o aggiornare un'EAA in qualsiasi momento (vedi *Gestione degli Attestati Elettronici* (pagina 100));
- DEVE garantire che l'EAA sia un elemento funzionale, affinché l'Utente possa accedere ai servizi forniti dai Verificatori di Attestati Elettronici in contesti digitali e di prossimità (vedi *Presentazione degli Attestati Elettronici* (pagina 67));
- DEVE mostrare nella Vista di Dettaglio un metodo di assistenza reso disponibile dalla Fonte Autentica tramite il parametro *data_capabilities.contacts '(vedi :ref:'functionalities: Assistenza Utente* e vedi *Registro delle Fonti Autentiche* (pagina 173)).

Di seguito un esempio di layout di Attestato Elettronico di Attributi, all'interno della Vista in Anteprima e Vista di Dettaglio.

5.5 Presentazione degli Attestati Elettronici

Il processo di presentazione permette all'Utente di accedere a un servizio oppure di dimostrare la titolarità di un dato o l'idoneità a effettuare una determinata azione. La presentazione degli Attestati Elettronici e la loro conseguente verifica, prevede l'interazione tra un'Istanza del Wallet, gestita dall'Utente, e un'Istanza di Relying Party (o App di Verifica). A seconda delle circostanze e del contesto di interazione si possono delineare i seguenti scenari:

- **Presentazione in prossimità**: l'Utente presenta i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici tramite l'Istanza del Wallet direttamente a un Verificatore di Attestati Elettronici o a un dispositivo preposto alla verifica in presenza;
- **Presentazione da remoto**: l'Utente presenta i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici tramite l'Istanza del Wallet ad un Verificatore di Attestati Elettronici predisposto per la verifica online al fine, ad esempio, di Autenticarsi e fruire dei servizi erogati.

Indipendentemente dal tipo di presentazione, il Fornitore di Wallet DEVE consentire all'Istanza del Wallet di informare l'Utente sull'identità della Relying Party. Nel caso di un Intermediario

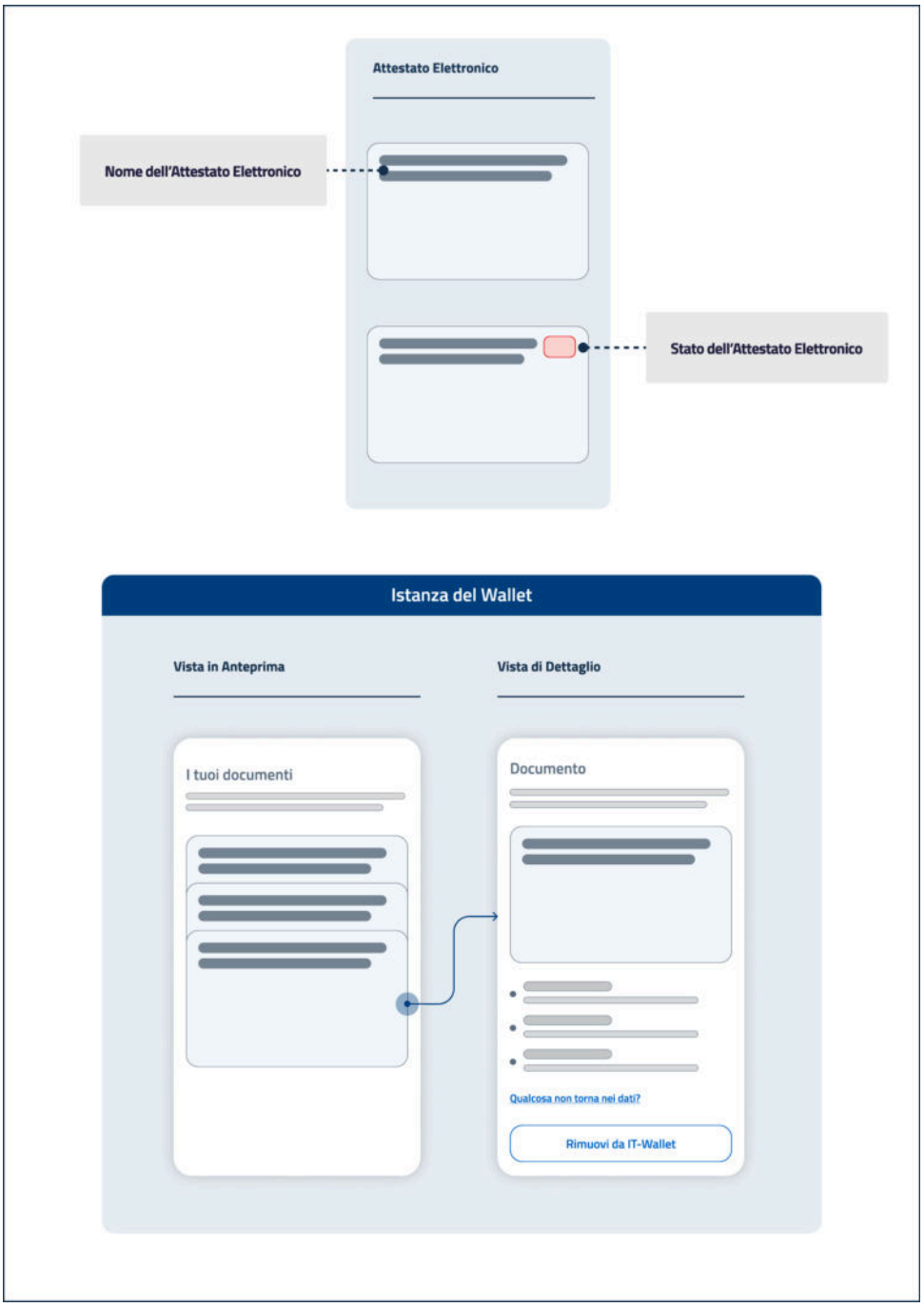


Fig. 5.16: Esempio di layout di Attestato Elettronico di Attributi, Vista in Anteprima e Vista di Dettaglio

di Relying Party, l'Utente DEVE essere informato del coinvolgimento dell'intermediario durante la fase di richiesta della presentazione. L'identità della Relying Party primaria DEVE essere sempre visibile all'Utente e NON DEVE essere sostituita da quella dell'Intermediario di Relying Party.

5.5.1 Presentazione in prossimità

La presentazione in prossimità consente all'Utente di esibire i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici tramite la propria Istanza del Wallet, secondo due diverse modalità:

- **Modalità supervisionata**: l'Utente, tramite l'Istanza del Wallet, presenta i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici, a un Verificatore di Attestati Elettronici (e.g. agente delle forze dell'ordine, operatore di sportello, etc.) dotato di un apposito sistema di verifica (*App di Verifica Mobile* (pagina 364));
- **Modalità non supervisionata**: l'Utente, tramite l'Istanza del Wallet, presenta i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici a un dispositivo preposto (e.g. tornello, totem, etc.) dotato di un apposito sistema di verifica (App di Verifica Embedded).

Di seguito i requisiti dell'Esperienza Utente relativi a entrambe le modalità che il Fornitori di Wallet DEVE garantire attraverso la propria Soluzione Wallet.

Modalità supervisionata

- L'Utente accede alla propria Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente accede alla funzionalità dedicata alla generazione del QR Code;
- L'Utente mostra il QR Code generato al soggetto che opera per conto del Verificatore di Attestati Elettronici, il quale provvede a scansionarlo tramite apposita app o sistema di verifica;
- L'Utente prende visione dei propri dati di identificazione personale e/o degli Attributi richiesti per la presentazione, del nome del Verificatore di Attestati Elettronici che li richiede e delle relative eventuali informative. L'Utente sceglie se presentare o meno eventuali dati non obbligatori ai fini della presentazione (Divulgazione Selettiva). L'Utente dà il proprio consenso per poter proseguire oppure annulla l'operazione;
- L'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente visualizza l'esito positivo della presentazione avvenuta.

In caso di errori nell'utilizzo dell'Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Modalità non supervisionata

- L'Utente accede alla propria Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente accede alla funzionalità dedicata alla generazione del QR Code;
- L'Utente mostra il QR Code generato al dispositivo preposto (ad esempio un tornello) del Verificatore di Attestati Elettronici per permetterne la scansione;
- L'Utente prende visione dei propri dati di identificazione personale e/o degli Attributi richiesti per la presentazione, del nome del Verificatore di Attestati Elettronici che li richiede e delle relative eventuali informative. L'Utente sceglie se presentare o meno eventuali dati non obbligatori ai fini della presentazione (Divulgazione Selettiva). L'Utente dà il proprio consenso per poter proseguire oppure annulla l'operazione;
- L'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente visualizza l'esito positivo della presentazione avvenuta.

In caso di errori nell'utilizzo dell'Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

5.5.2 Presentazione da remoto

La presentazione da remoto consente all'Utente di esibire i propri dati di identificazione personale e/o un set di Attributi contenuti in uno o più Attestati Elettronici, facendo interagire la propria Istanza del Wallet con il Touchpoint di un Verificatore di Attestati Elettronici, tramite l'apposito *Engagement Button* (pagina 53).

Tale presentazione può avvenire in due diverse modalità, sulla base del tipo di dispositivo utilizzato per accedere al servizio di interesse:

- **Same-device**: nel caso in cui l'Utente voglia accedere a un servizio digitale online integrato con un apposito sistema di verifica (*App di Verifica Web* (pagina 371)) utilizzando lo stesso dispositivo su cui ha installato l'Istanza del Wallet;
- **Cross-device**: nel caso in cui l'Utente voglia accedere a un servizio digitale integrato con un apposito sistema di verifica (*App di Verifica Web* (pagina 371)) utilizzando un dispositivo differente rispetto a quello su cui ha installato l'Istanza del Wallet.



Fig. 5.17: Esempio di Esperienza Utente nella presentazione in prossimità

Di seguito i requisiti dell'Esperienza Utente relativi a entrambe le modalità che il Fornitore di Wallet DEVE garantire attraverso la propria Soluzione Wallet.

Same-device

- L'Utente clicca l'*Engagement Button* (pagina 53) reso disponibile nel Touchpoint del Verificatore di Attestati Elettronici;
- L'Utente seleziona la Soluzione Wallet con la quale procedere, attraverso un'interfaccia che DEVE seguire le indicazioni e le funzionalità previste per la Selection Page descritta nella sezione *Autenticazione* (pagina 80);
- L'Utente accede all'Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente prende visione dei propri dati di identificazione personale e/o degli Attributi richiesti per la presentazione, del nome del Verificatore di Attestati Elettronici che li richiede e delle relative eventuali informative. L'Utente sceglie se presentare o meno eventuali dati non obbligatori ai fini della presentazione (Divulgazione Selettiva). L'Utente dà il proprio consenso per poter proseguire oppure annulla l'operazione;
- L'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente visualizza nell'Istanza del Wallet l'esito positivo della presentazione avvenuta;
- L'Utente torna al flusso nel Touchpoint del Verificatore di Attestati Elettronici su cui visualizza l'esito della presentazione completata.

In caso di errori nell'utilizzo dell'Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

Cross-device

- L'Utente clicca l'*Engagement Button* (pagina 53) reso disponibile nel Touchpoint del Verificatore di Attestati Elettronici che l'Utente sta navigando da un dispositivo diverso da quello su cui è installata l'Istanza del Wallet;
- L'Utente seleziona la Soluzione Wallet con la quale procedere, attraverso un'interfaccia che DEVE seguire le indicazioni e le funzionalità previste per la Selection Page descritta nella sezione *Autenticazione* (pagina 80);
- L'Utente inquadra il QR Code reso disponibile dal Verificatore di Attestati Elettronici, utilizzando la sua Istanza del Wallet o la fotocamera del proprio dispositivo; l'interfaccia del QR Code DEVE seguire le indicazioni e le funzionalità previste per la QR Code Page descritta nella sezione *Autenticazione* (pagina 80);

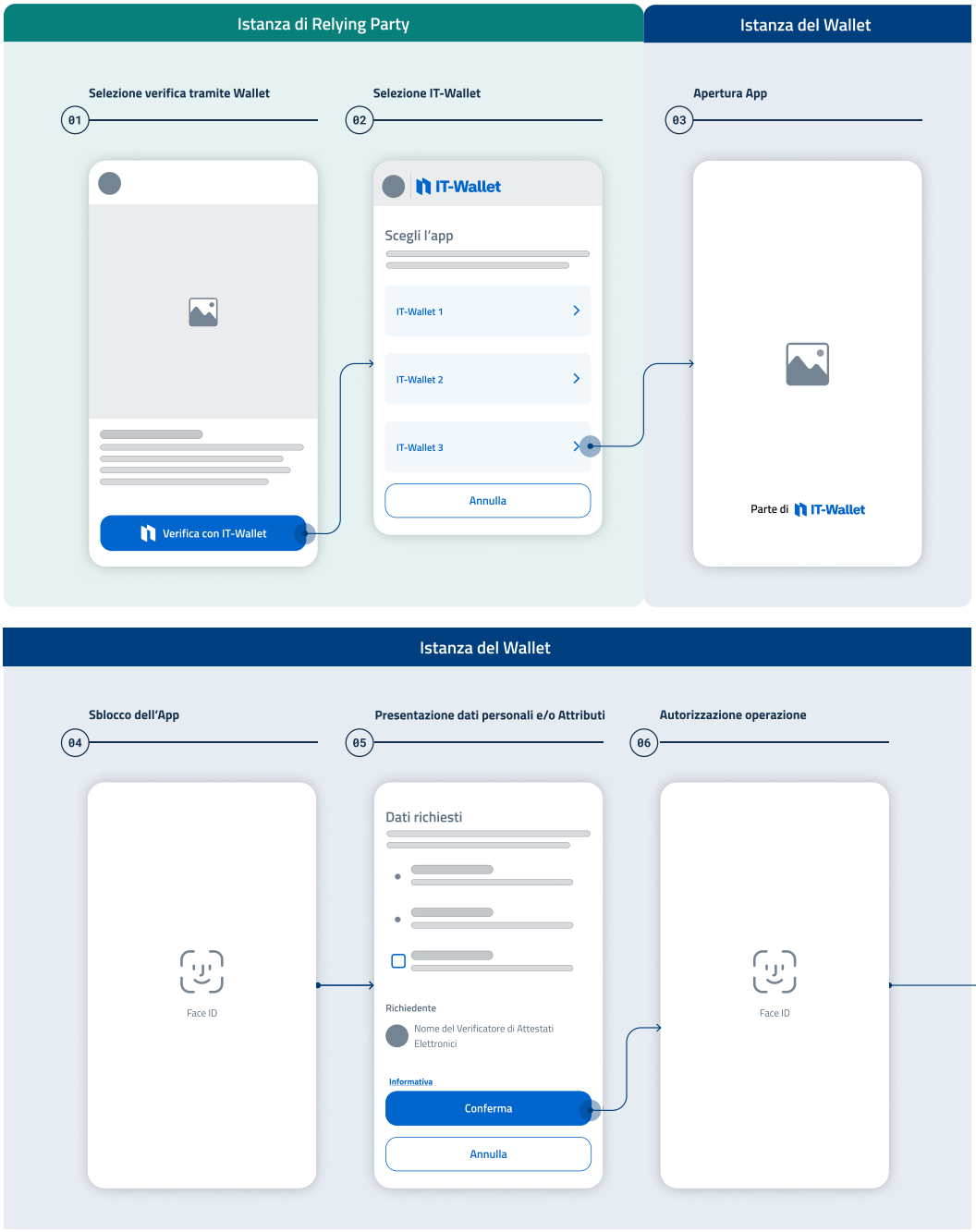


Fig. 5.18: Esempio di Esperienza Utente nella presentazione da remoto, same-device - 01

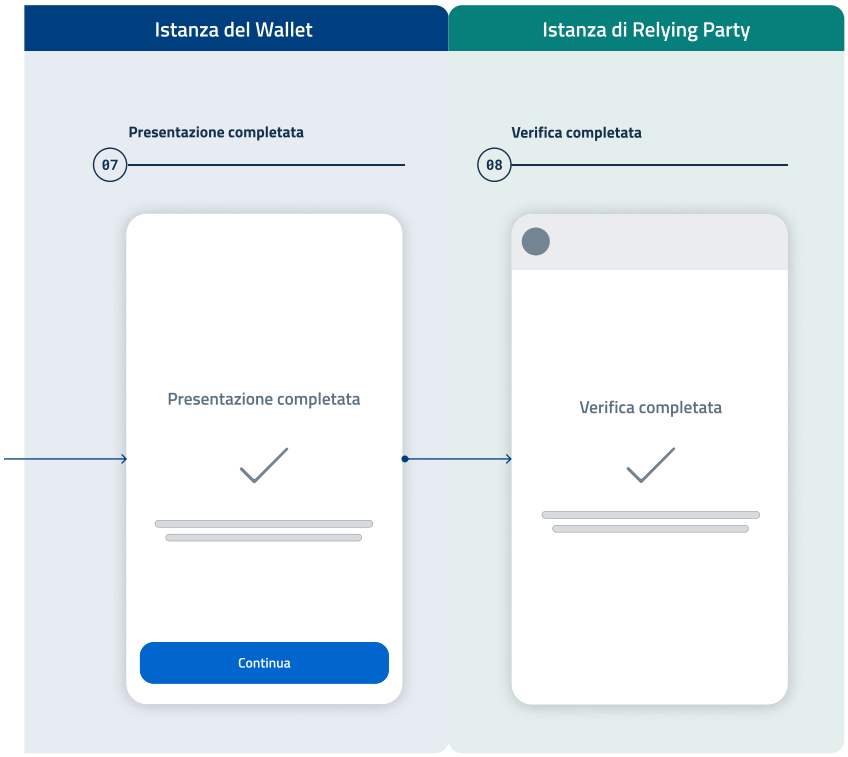


Fig. 5.19: Esempio di Esperienza Utente nella presentazione da remoto, same-device - 02

- L'Utente prende visione dei propri dati di identificazione personale e/o degli Attributi richiesti per la presentazione, del nome del Verificatore di Attestati Elettronici che li richiede e delle relative eventuali informative. L'Utente sceglie se presentare o meno eventuali dati non obbligatori ai fini della presentazione (Divulgazione Selettiva). L'Utente dà il proprio consenso per poter proseguire oppure annulla l'operazione;
- L'Utente autorizza l'operazione utilizzando la modalità di sblocco precedentemente impostata;
- L'Utente visualizza nell'Istanza del Wallet l'esito positivo della presentazione avvenuta;
- L'Utente torna sul Touchpoint del Verificatore di Attestati Elettronici e visualizza l'esito della presentazione completata.

In caso di errori nell'utilizzo dell'Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.

Di seguito è rappresentata a titolo esemplificativo la pagina con l'*Engagement Button* (pagina 53), oltre alla Selection Page, alla QR Code Page e alla Thank You Page con gli elementi dell'interfaccia e i testi adattati al flusso di *Presentazione da remoto* (pagina 70).

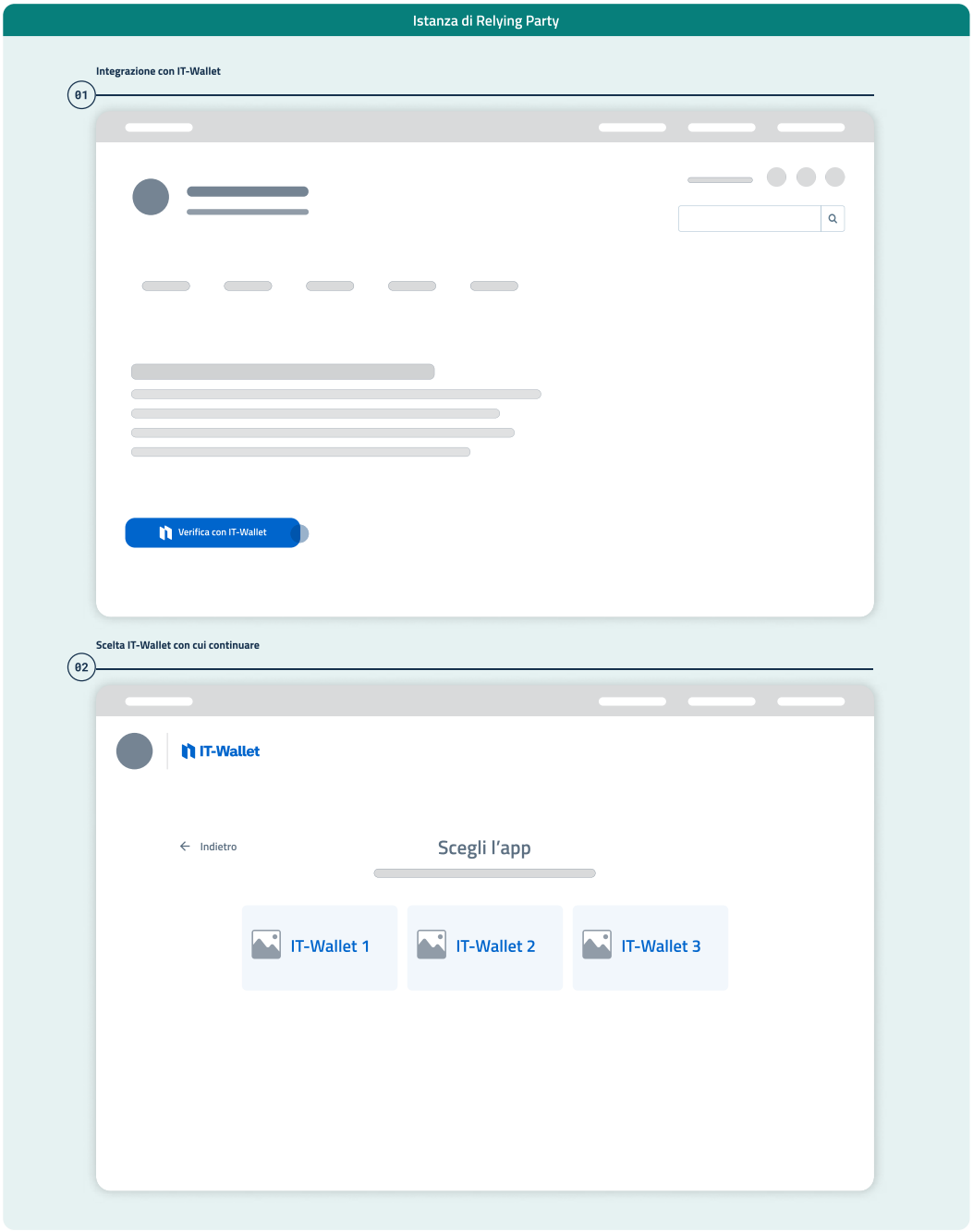


Fig. 5.20: Esempio di Esperienza Utente nella presentazione da remoto, cross-device - 01

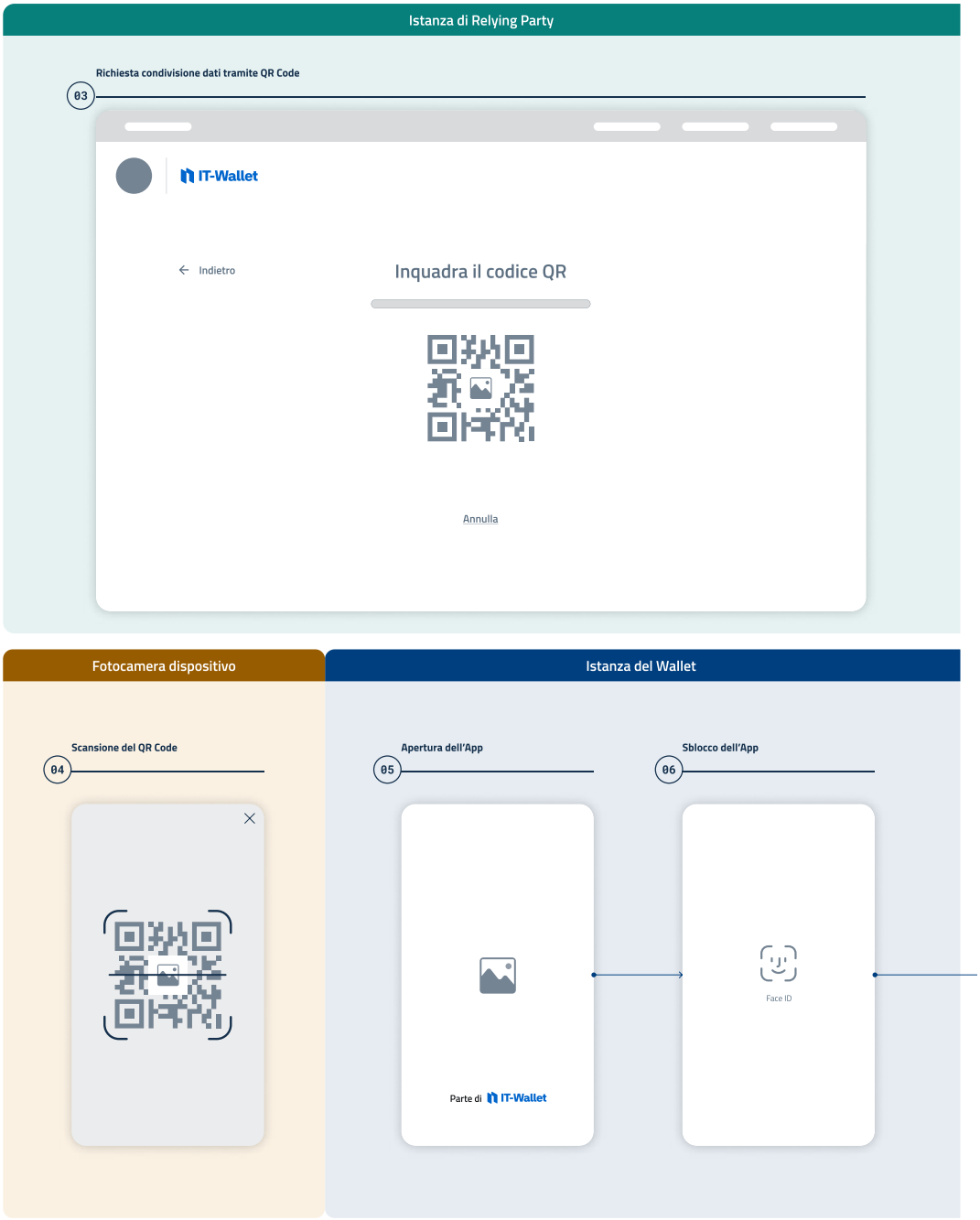


Fig. 5.21: Esempio di Esperienza Utente nella presentazione da remoto, cross-device - 02

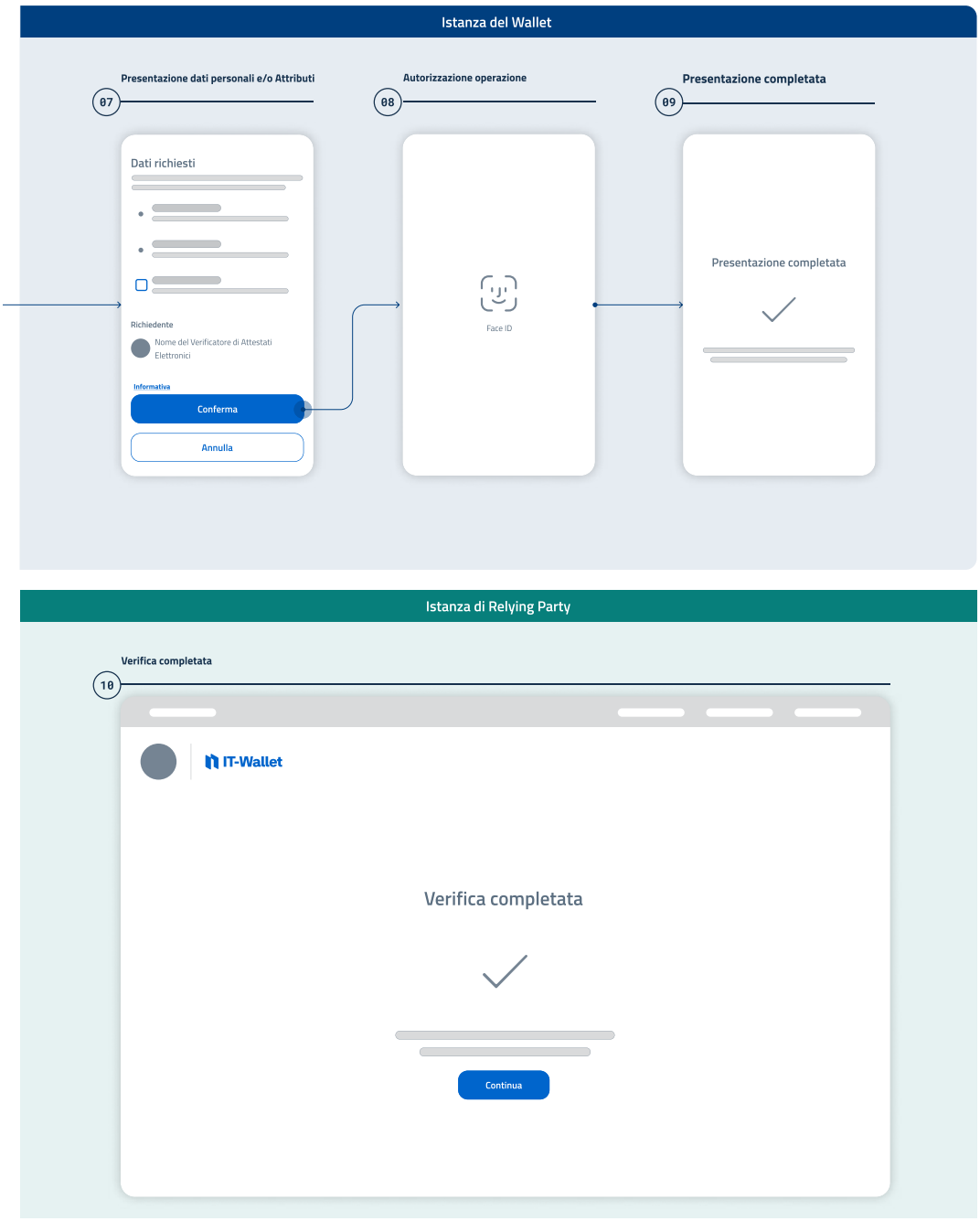


Fig. 5.22: Esempio di Esperienza Utente nella presentazione da remoto, cross-device - 03

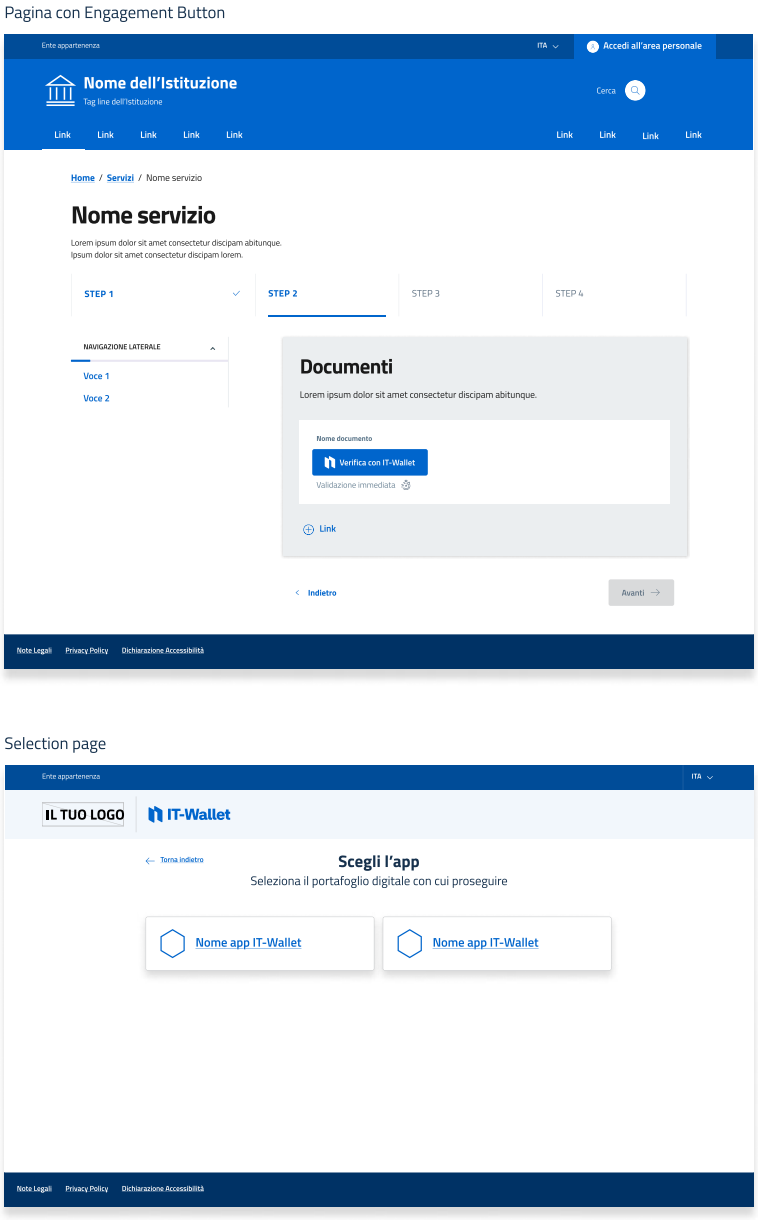
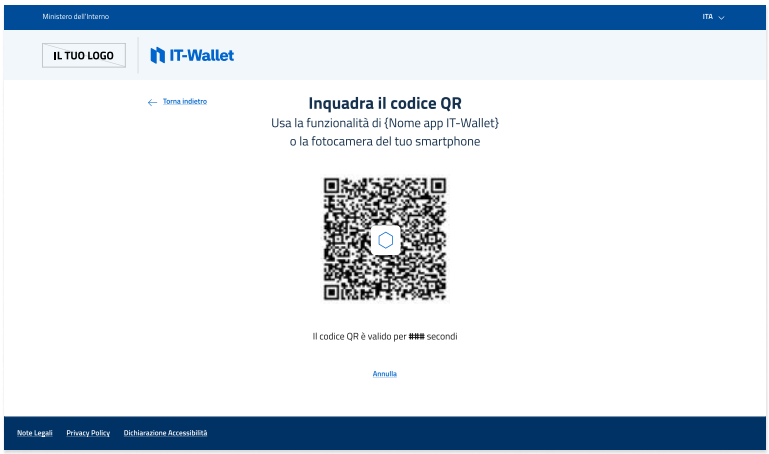


Fig. 5.23: Esempio di pagina con Engagement Button, Selection page, QR code page, Thank you page per la presentazione da remoto

QR code page



Thank you page

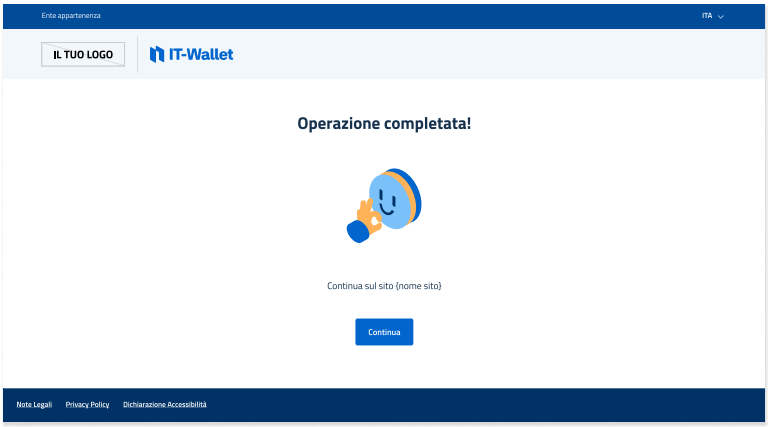


Fig. 5.24: Esempio di pagina con Engagement Button, Selection page, QR code page, Thank You Page per la presentazione da remoto

Autenticazione

L'Autenticazione è un caso d'uso specifico della presentazione remota e consente all'Utente di accedere in modo sicuro ai servizi resi disponibili da Verificatori di Attestati Elettronici pubblici e privati, presentando i propri dati di identificazione personale ed eventualmente un set di Attributi contenuti negli Attestati Elettronici di Attributi ottenuti. Questo garantisce all'Utente il pieno controllo sui propri dati e la possibilità di condividere anche i soli dati strettamente necessari alla verifica da parte del Verificatore di Attestati Elettronici.

Il processo di Autenticazione può avvenire in entrambe le modalità same-device e cross-device descritte sopra. Per quanto riguarda i requisiti funzionali a supporto dell'Esperienza Utente, si DEVONO rispettare gli stessi requisiti previsti per la *presentazione in remoto* (pagina 70) nelle due modalità (same-device e cross-device).

Infatti, a livello di Esperienza Utente, il processo di Autenticazione si distingue da un generico flusso di presentazione principalmente per le modalità di avvio del processo, in questo caso reso possibile a partire da uno specifico pulsante, l'*Authentication Button* (pagina 96).

Al fine di garantire un processo di Autenticazione adeguato e coerente tra tutti i Verificatori di Attestati Elettronici, ciascun Verificatore di Attestati Elettronici DEVE rispettare i requisiti relativi all'aspetto grafico e all'Esperienza Utente descritti di seguito, unitamente al rispetto di [RIF_ACCESSIBILITÀ] e, nel caso di enti pubblici, delle [LG_DESIGN].

Entrambi i flussi sono rappresentati di seguito con wireframe esemplificativi.

I Verificatori di Attestati Elettronici DOVREBBERO utilizzare le official-resources:Risorse Ufficiali per la progettazione. Qualora non intendano utilizzare tali risorse open source, i Verificatori di Attestati Elettronici POSSONO sviluppare in autonomia le Soluzioni Tecniche abilitanti il flusso di Autenticazione, assicurando coerenza con le specifiche fornite di seguito.

Nota

Le immagini presenti in questa sezione sono da considerarsi esemplificative in quanto oggetto di evolutive di interfaccia (UI).

I Verificatori di Attestati Elettronici, in ogni caso, DEVONO abilitare il processo di Autenticazione rendendo disponibili le seguenti pagine:

- **Discovery Page**: ha l'obiettivo di mostrare all'Utente tutti i metodi di Autenticazione disponibili;
- **Selection Page**: ha lo scopo di mostrare all'Utente tutte le Soluzioni Wallet presenti nel Registro del Sistema IT-Wallet e permettere di scegliere con quale continuare il processo di Autenticazione;

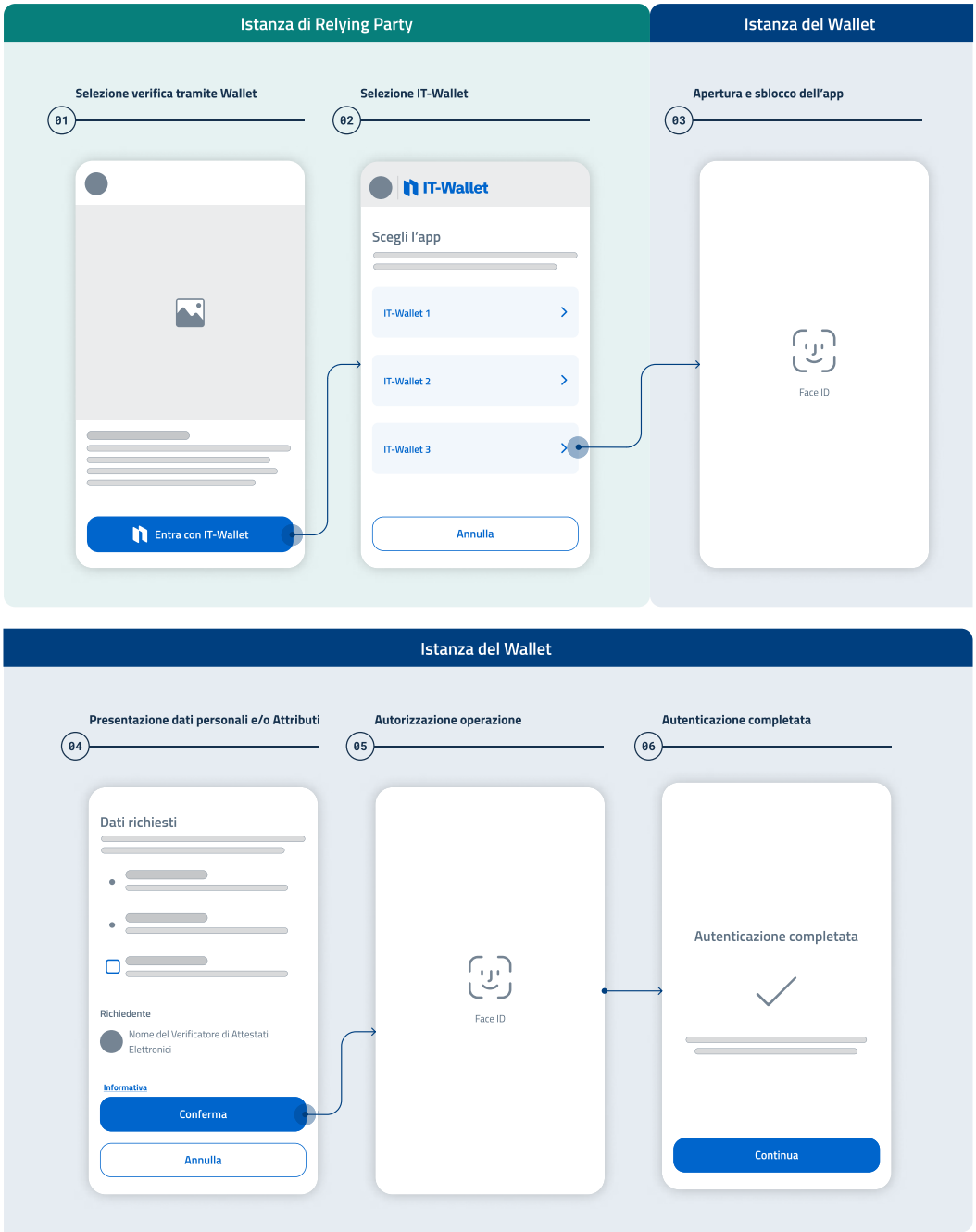


Fig. 5.25: Esempio di Esperienza Utente di Autenticazione same-device - 01

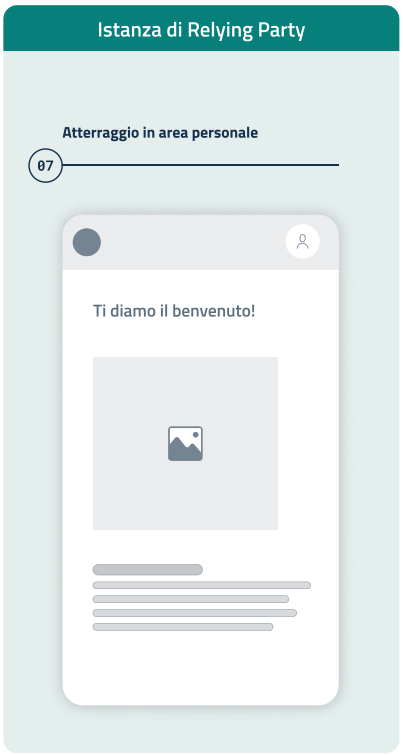


Fig. 5.26: Esempio di Esperienza Utente di Autenticazione same-device - 02

- **QR Code Page** (*solo per modalità cross-device*): ha lo scopo di invitare l'Utente a inquadrare il codice QR;
- **Thank You Page**: ha lo scopo di comunicare all'Utente l'avvenuta Autenticazione;
- **Error Page**: ha lo scopo di comunicare all'Utente eventuali errori legati al flusso di Autenticazione.

Tali pagine DEVONO prevedere i seguenti elementi trasversali ricorrenti, in continuità con l'Identità Visiva del Touchpoint del Verificatore di Attestati Elettronici:

- un **header e/o un subheader**, che permette all'Utente di tornare alla pagina precedente;
- un **footer** che include l'informativa privacy, le note legali e la Dichiarazione di Accessibilità, ove previsto da normativa.

Di seguito invece gli elementi specifici caratteristici delle diverse pagine.

Discovery Page

Per garantire l'Autenticazione tramite il Sistema IT-Wallet, il Verificatore di Attestati Elettronici PUÒ aggiornare la propria Discovery Page con quella resa disponibile nelle official-resources: Risorse Ufficiali.

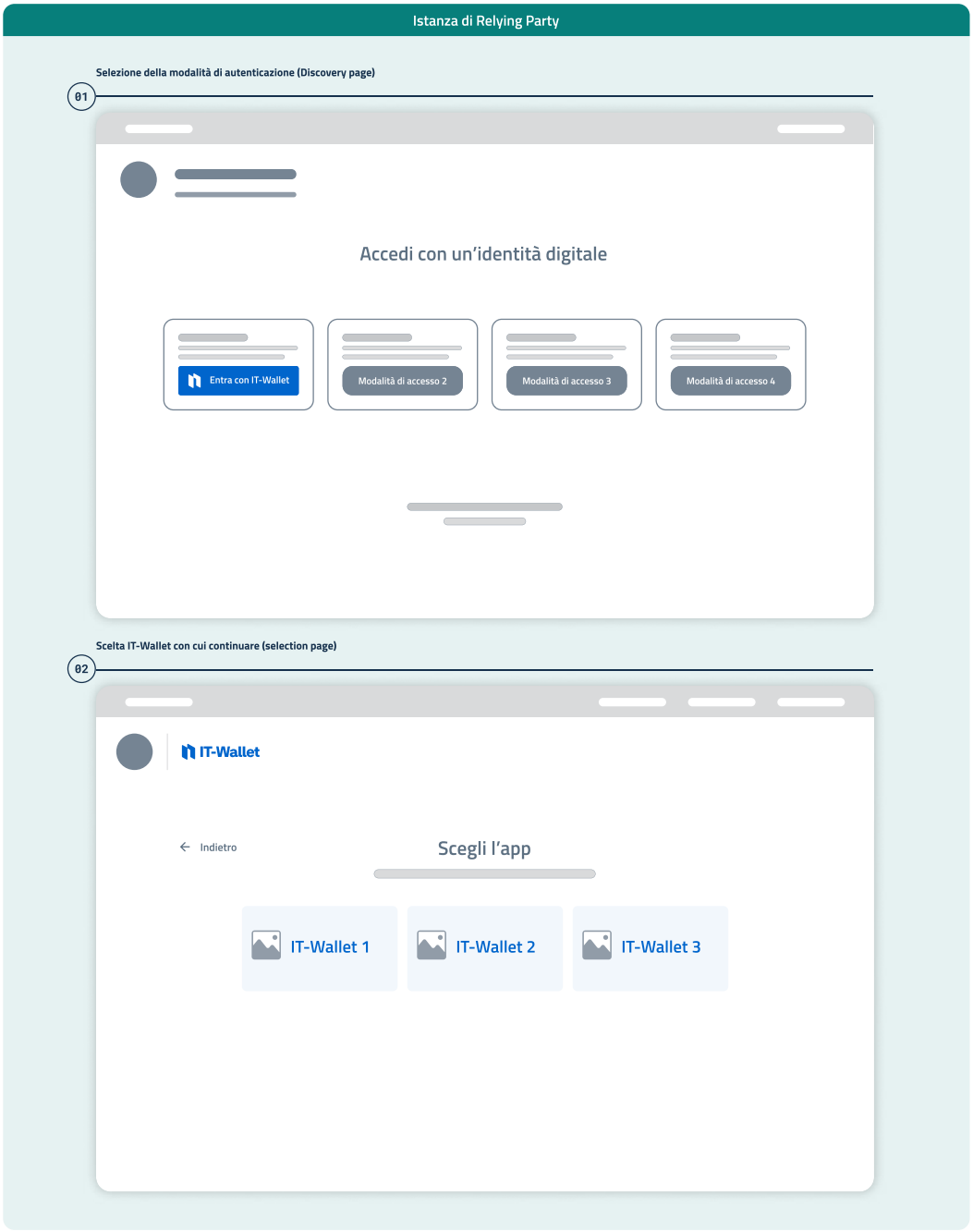


Fig. 5.27: Esempio di Esperienza Utente di Autenticazione cross-device - 01

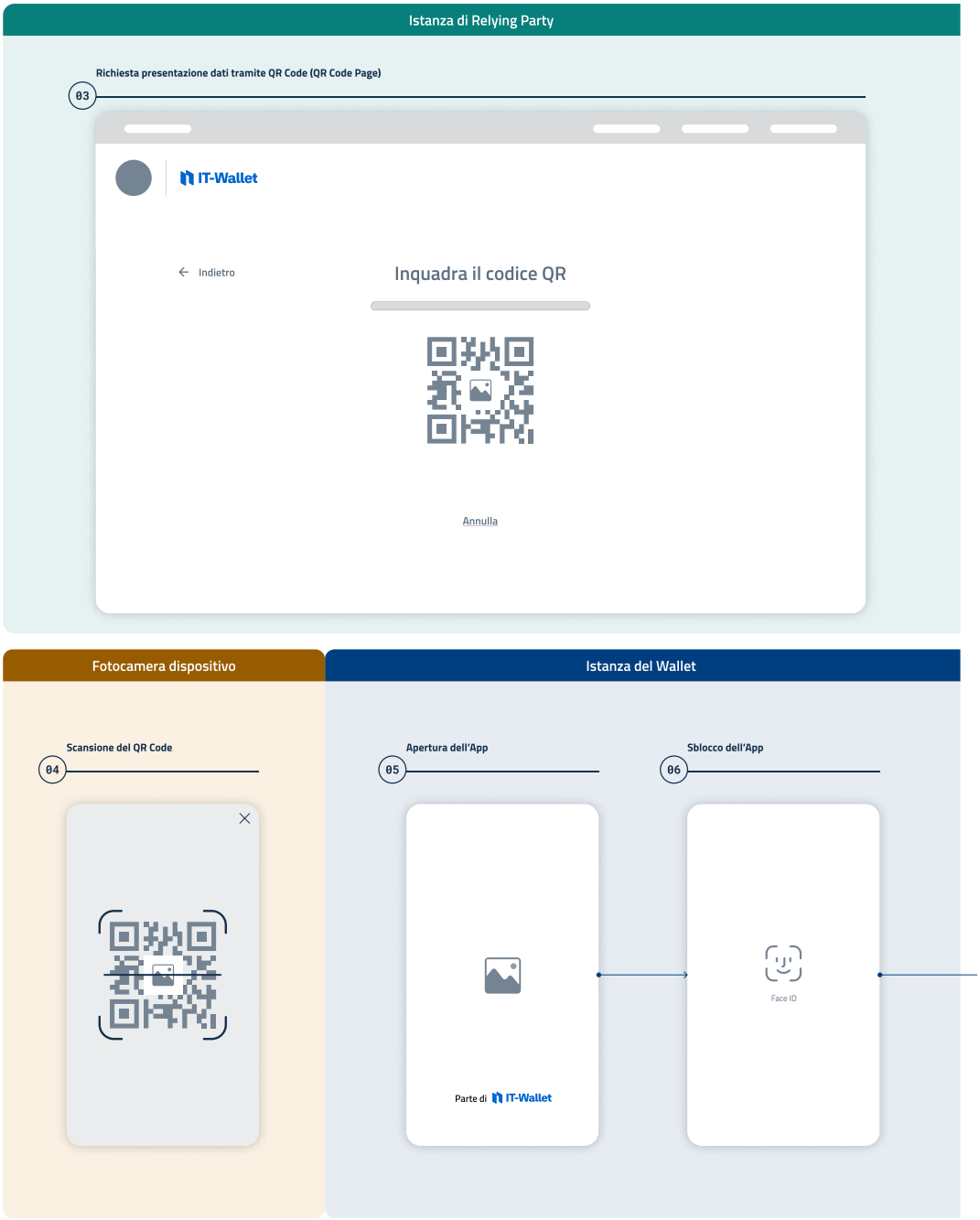


Fig. 5.28: Esempio di Esperienza Utente di Autenticazione cross-device - 02

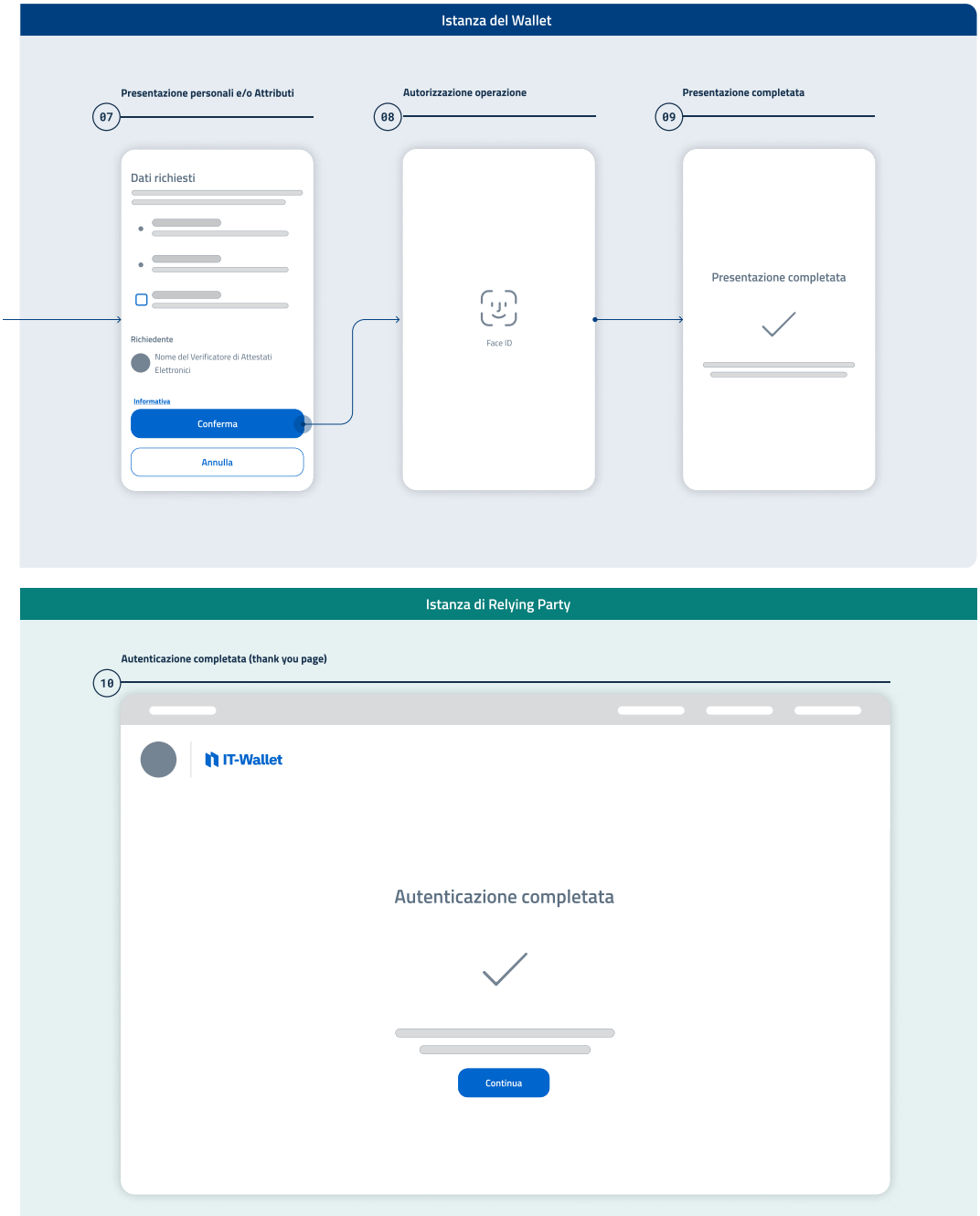


Fig. 5.29: Esempio di Esperienza Utente di Autenticazione cross-device - 03

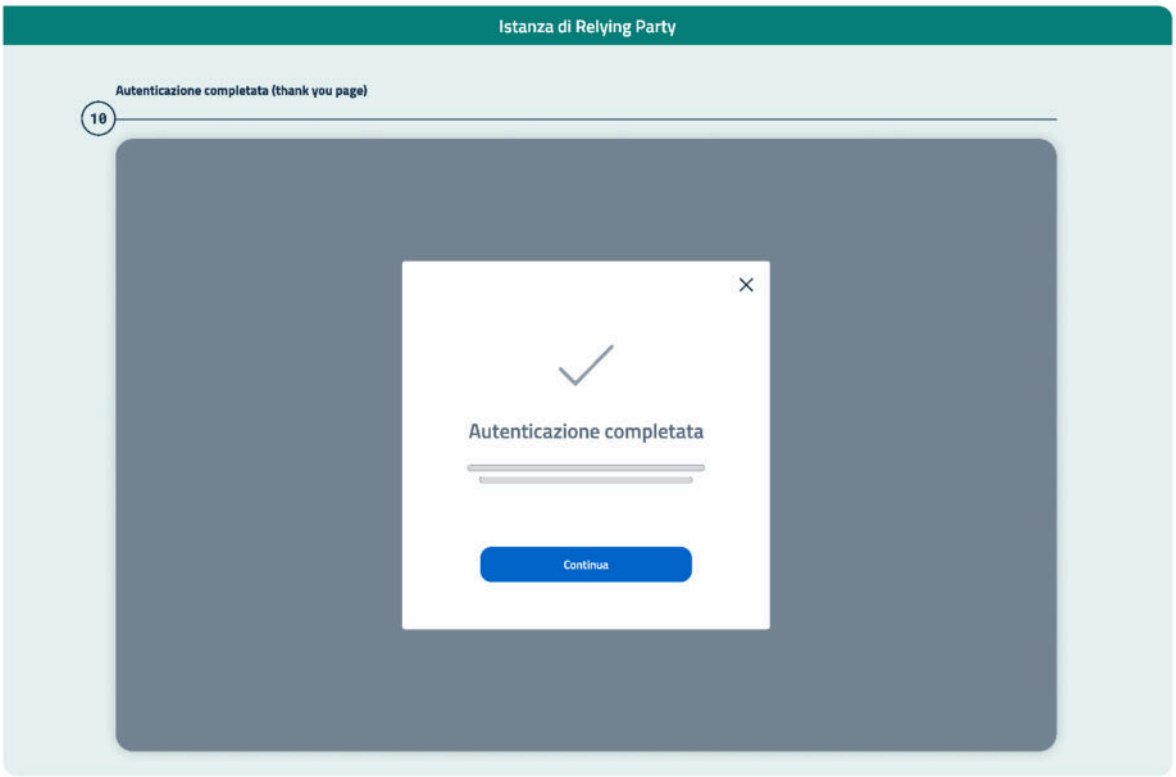


Fig. 5.30: Esempio di Esperienza Utente di Autenticazione cross-device - 04

In alternativa, il Verificatore di Attestati Elettronici PUÒ mantenere la propria Discovery Page, ma DEVE in ogni caso integrare l'Authentication Button, come da indicazioni presenti nella sezione *Authentication Button* (pagina 96).

Il Verificatore di Attestati Elettronici che implementa la pagina:

- DEVE garantire la presenza di tutte le modalità di Autenticazione tramite l'identità digitale tra cui la modalità di Autenticazione del Sistema IT-Wallet, quindi tramite l'Authentication Button;
- PUÒ presentare anche modalità di Autenticazione alternative, se disponibili;
- DOVREBBE garantire informazioni minime a supporto, per permettere all'Utente di compiere una scelta consapevole e informata.

Nel caso l'Utente stia navigando la pagina del Verificatore di Attestati Elettronici da un Touchpoint diverso da quello su cui ha attivato l'Istanza del Wallet (modalità cross-device), la scelta di Autenticazione tramite il Sistema IT-Wallet DEVE condurre l'Utente alla QR Code Page.

Nel caso in cui invece l'Utente stia navigando la pagina del Verificatore di Attestati Elettronici dallo stesso Touchpoint su cui ha attivato l'Istanza del Wallet (modalità same-device) tale pagina DEVE condurre l'Utente all'apertura della propria Istanza del Wallet.

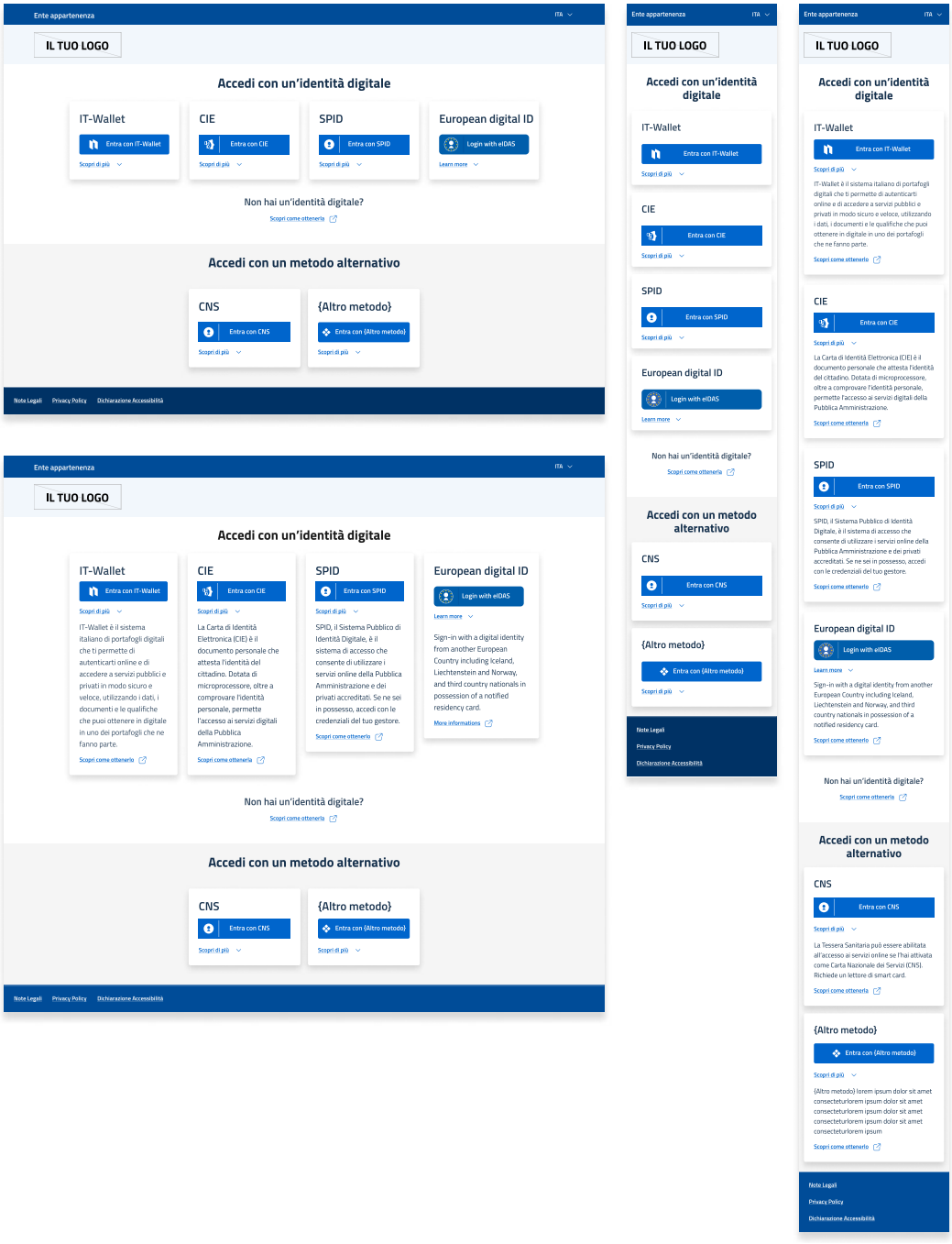


Fig. 5.31: Modello di layout di Discovery Page a griglia

Selection Page

La Selection Page è la pagina su cui atterra l'Utente dopo che ha scelto di Autenticarsi tramite il Sistema IT-Wallet, e ha lo scopo di presentare all'Utente le Soluzioni Wallet disponibili per effettuare l'Autenticazione.

Nota

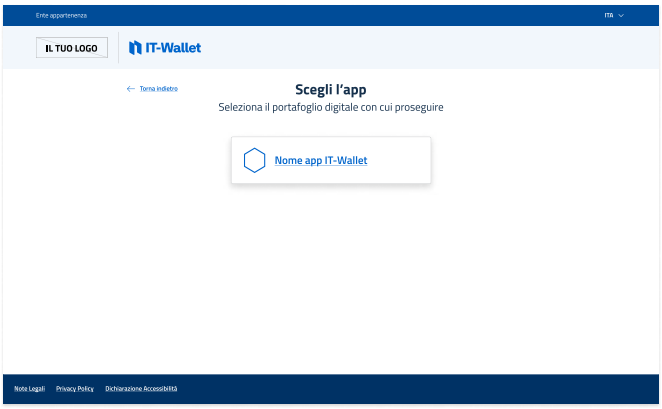
Questa sezione descrive come visualizzare la Selection Page come parte di un processo di Autenticazione. La stessa pagina DOVREBBE essere utilizzata anche dai Verificatori di Attestati Elettronici durante la presentazione e da terze parti che supportano il Credential Offer per abilitare l'opzione di selezione della Soluzione Wallet. Ulteriori dettagli sono forniti in *Flusso Remoto* (pagina 509) e *Flusso Credential Offer* (pagina 477).

Il Verificatore di Attestati Elettronici DEVE implementare la Selection Page resa disponibile nelle official-resources:Risorse Ufficiali.

Il Verificatore di Attestati Elettronici che implementa la pagina:

- DEVE includere gli elementi propri dell'Identità Visiva del Sistema IT-Wallet, tra cui il Logo affiancandolo al proprio logo secondo le indicazioni fornite nella sezione *Identità Visiva* (pagina 16);
- DEVE assicurare che i copy presenti nella pagina rispecchino quelli riportati nelle official-resources:Risorse Ufficiali;
- DEVE presentare ogni Soluzione Wallet presente nel Registro del Sistema IT-Wallet attraverso un componente modulare che mostra il logo e il nome per esteso recuperati come descritto in *Flusso di Recupero dei Wallet Metadata* (pagina 570);
- DEVE presentare le Soluzioni Wallet in un layout dinamico che si adatta al numero di Soluzioni Wallet disponibili: quando il numero di Solutioni Wallet è inferiore a 2, la Selection Page DEVE distribuire le Soluzioni Wallet all'interno di un layout ad una colonna centrale. Altrimenti, quando il numero di Soluzioni Wallet è pari o superiore a 3, la Selection Page DEVE distribuire le Soluzioni Wallet in una griglia a 2 colonne; in ogni caso DEVE essere garantito un ordinamento randomico;
- DEVE permettere all'Utente di cercare una Soluzione Wallet attraverso una funzionalità di filtro per nome, quando presenti più di 5 Soluzioni Wallet;
- DEVE permettere all'Utente di scoprire, in caso di necessità, quali sono le Soluzioni Wallet presenti nel Registro del Sistema IT-Wallet, predisponendo un rimando al sito ufficiale del Sistema IT-Wallet;

Selection page | 1 Soluzione Wallet



Selection page | < 3 Soluzioni Wallet

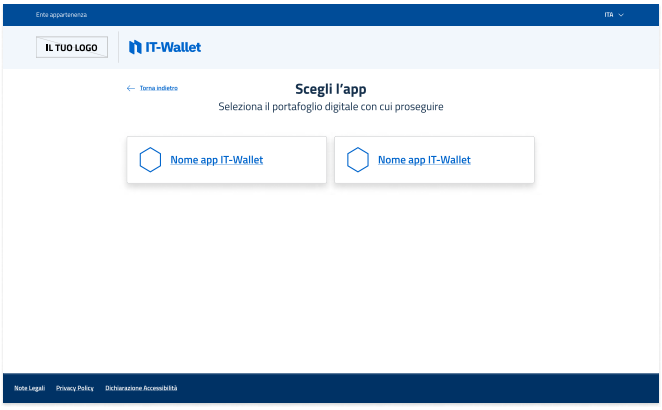


Fig. 5.32: Selection Page desktop

Selection page | + 3 Soluzioni Wallet

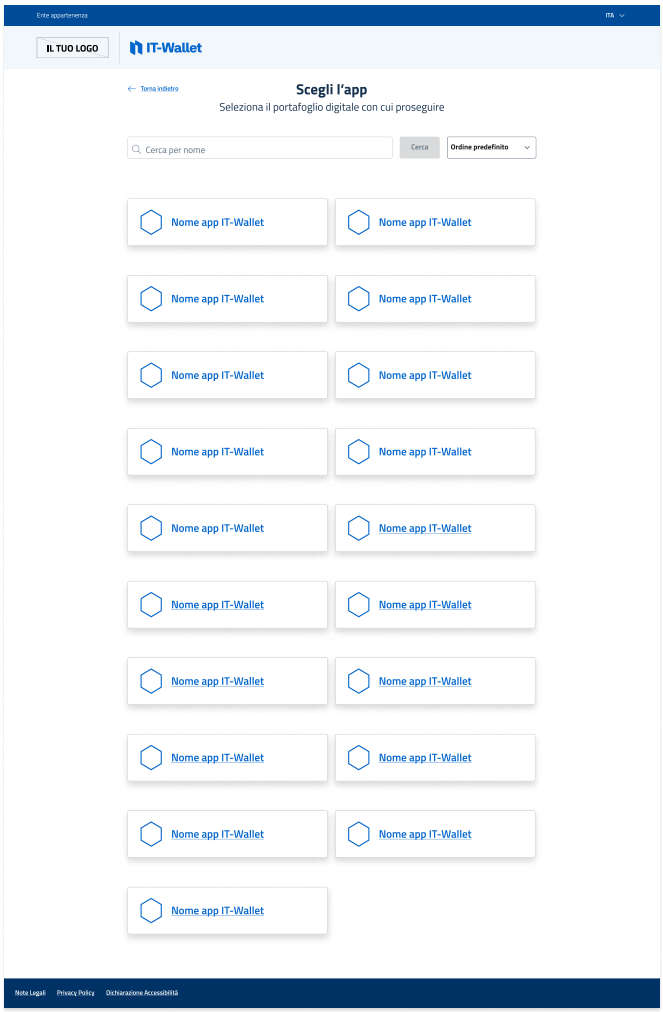


Fig. 5.33: Selection Page desktop

Selection page | Ordinamento Soluzioni

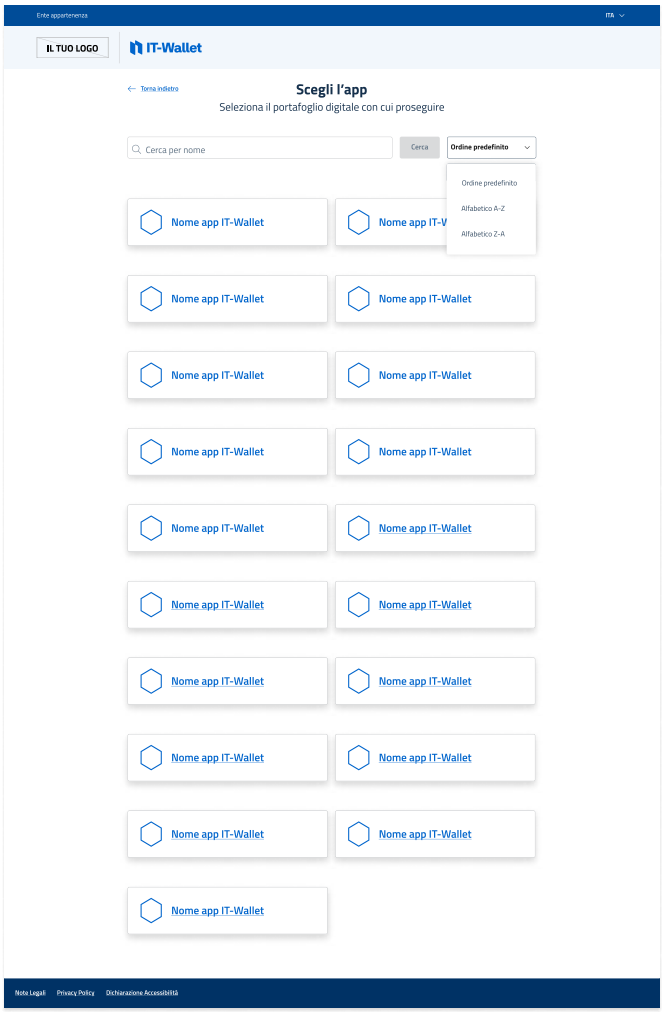


Fig. 5.34: Selection Page desktop

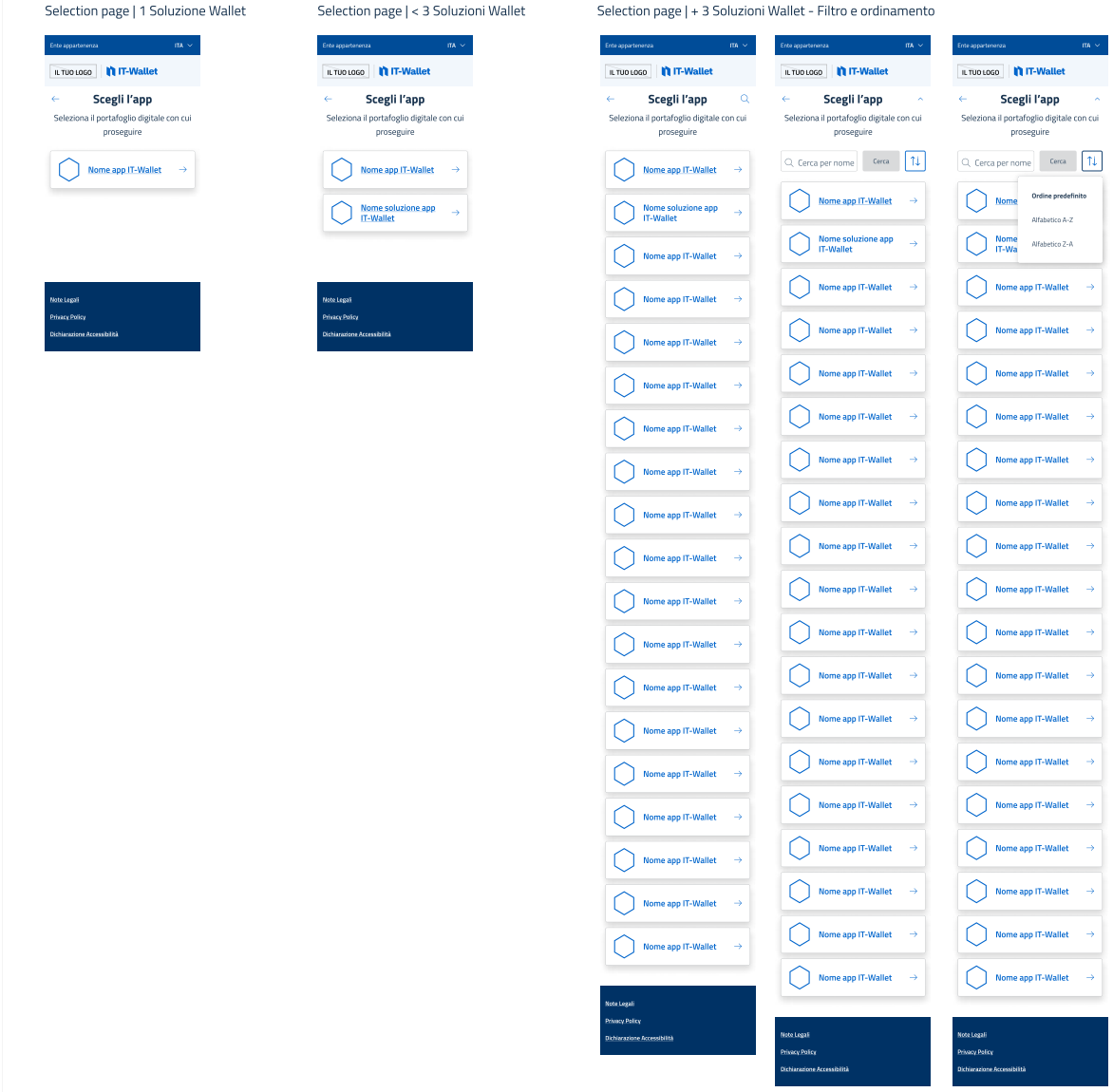


Fig. 5.35: Selection Page mobile

- DEVE includere una Call to Action che permetta all'Utente di interrompere l'operazione e tornare alla pagina precedente (ad esempio la Discovery Page nel caso di un processo di Autenticazione).

Il Verificatore di Attestati Elettronici PUÒ inserire un componente testuale per promuovere la modalità di Autenticazione tramite IT-Wallet, che rimandi al sito ufficiale del Sistema, come rappresentato nelle official-resources:Risorse Ufficiali.

QR Code Page (*solo per modalità cross-device*)

La QR Code Page è la pagina su cui atterra l'Utente che ha scelto l'Autenticazione tramite il Sistema IT-Wallet in un flusso cross-device, e ha lo scopo di invitare l'Utente a scannerizzare, con la propria Istanza del Wallet o con la fotocamera del proprio dispositivo, il codice QR generato.

Il Verificatore di Attestati Elettronici DEVE implementare la QR Code Page (flusso cross-device) resa disponibile nelle official-resources:Risorse Ufficiali.



Fig. 5.36: QR Code Page

Il Verificatore di Attestati Elettronici che implementa la pagina:

- DEVE includere gli elementi propri dell'Identità Visiva del Sistema IT-Wallet, tra cui il Logo;
- DEVE assicurare che i copy presenti nella pagina rispecchino quelli riportati nelle official-resources:Risorse Ufficiali;
- DEVE includere una Call to Action che, se prevista la scadenza del codice QR, permetta all'Utente di generare un nuovo codice QR;

- DEVE includere una Call to Action che permetta all'Utente di annullare l'operazione e tornare alla Discovery Page.

Inoltre, nel rispetto di [RIF_ACCESSIBILITÁ], relativamente al codice QR, il Verificatore di Attestati Elettronici:

- DEVE rispettare le dimensioni minime raccomandate per garantire una scansione efficace. Una misura di 150x150 pixel è generalmente adeguata, ma per codici con alta densità di dati (e.g. URL lunghi o numerosi caratteri), è consigliabile aumentarla a 300x300 pixel o più;
- DEVE garantire un contrasto minimo tra il codice QR e lo sfondo (la condizione ideale prevede uno sfondo bianco con un codice QR nero);
- DEVE evitare inversioni di colore tra sfondo e codice QR;
- DEVE limitare la presenza a un solo codice QR per pagina;
- DEVE garantire nitidezza e alta qualità;
- DEVE garantire il formato SVG;
- DEVE garantire che non venga parzialmente nascosto da testo o altri elementi.

Thank You Page

La Thank You Page è la pagina sui cui l'Utente atterra una volta concluso il processo di Autenticazione attraverso la propria Istanza del Wallet e ha l'obiettivo di informare l'Utente sull'avvenuta Autenticazione.

Il Verificatore di Attestati Elettronici DEVE implementare la Thank You Page resa disponibile nelle official-resources:Risorse Ufficiali.

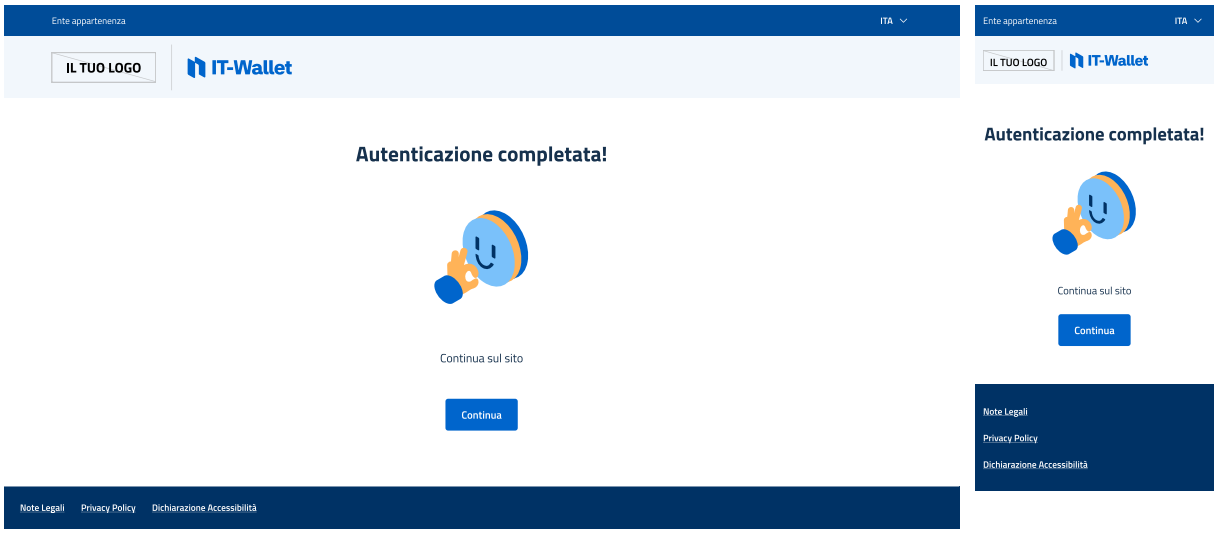


Fig. 5.37: Thank You Page

Il Verificatore di Attestati Elettronici che implementa la pagina:

- DEVE includere gli elementi propri dell'Identità Visiva del Sistema IT-Wallet, tra cui il Logo e un'icona o altro elemento grafico che aiuti a veicolare il messaggio della pagina;
- DEVE assicurare che i copy presenti nella pagina rispecchino quelli riportati nelle official-resources:Risorse Ufficiali;
- DEVE prevedere una Call to Action che inviti l'Utente a proseguire nell'area riservata del Verificatore di Attestati Elettronici.

Error Page

La pagina di errore rappresenta quella tipologia di pagina su cui l'Utente atterra in caso di errori nel corso del flusso di Autenticazione, e ha lo scopo di comunicare all'Utente la natura di tali errori (es. errore tecnico, assenza di rete, malfunzionamento dell'Istanza del Wallet, consenso alla presentazione dei dati negato etc.) e di presentare le azioni che l'Utente può intraprendere. Per approfondimenti sulle casistiche di errore si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il Verificatore di Attestati Elettronici DEVE implementare la Error Page resa disponibile nelle official-resources:Risorse Ufficiali.

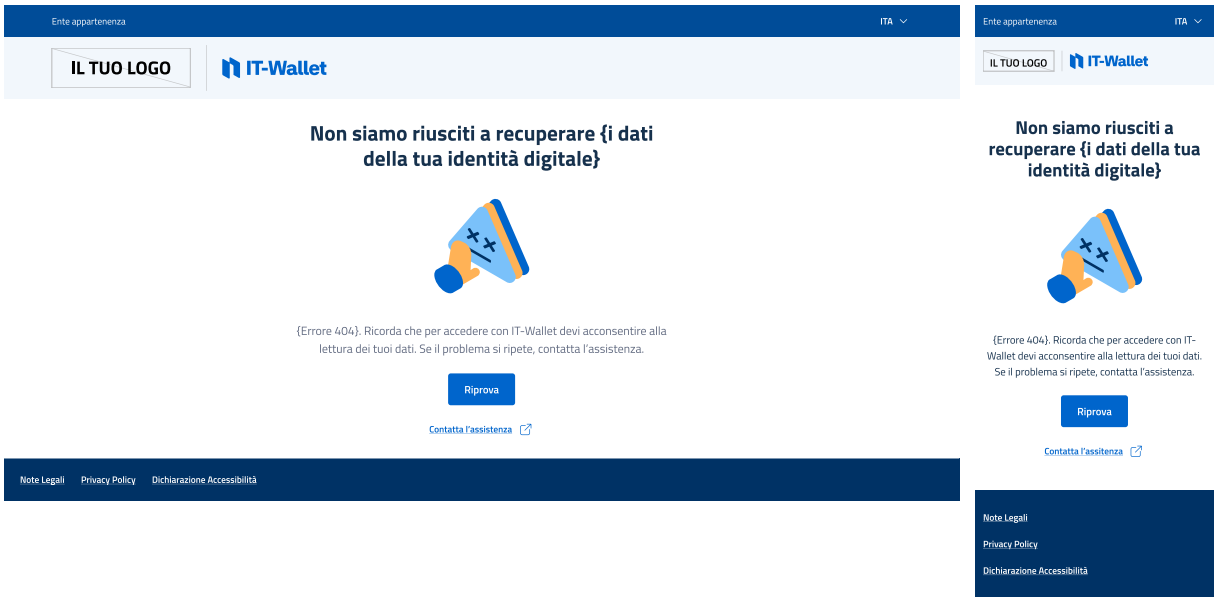


Fig. 5.38: Error Page

Il Verificatore di Attestati Elettronici che implementa la pagina:

- DEVE includere gli elementi propri dell'Identità Visiva del Sistema IT-Wallet, tra cui il Logo e un'icona o altro elemento grafico che aiuti a veicolare la natura dell'errore;
- DEVE assicurare che i copy presenti nella pagina rispecchino quelli riportati nelle official-resources:Risorse Ufficiali;

- DEVE prevedere una o più Call to Action che invitino l'Utente a intraprendere le azioni previste (es. riprova, contatta l'assistenza, etc.).

Authentication Button

L'Authentication Button "Entra con IT-Wallet" funge da *Engagement Button* (pagina 53), fornendo agli Utenti un modo standardizzato per Autenticarsi utilizzando il proprio portafoglio digitale.

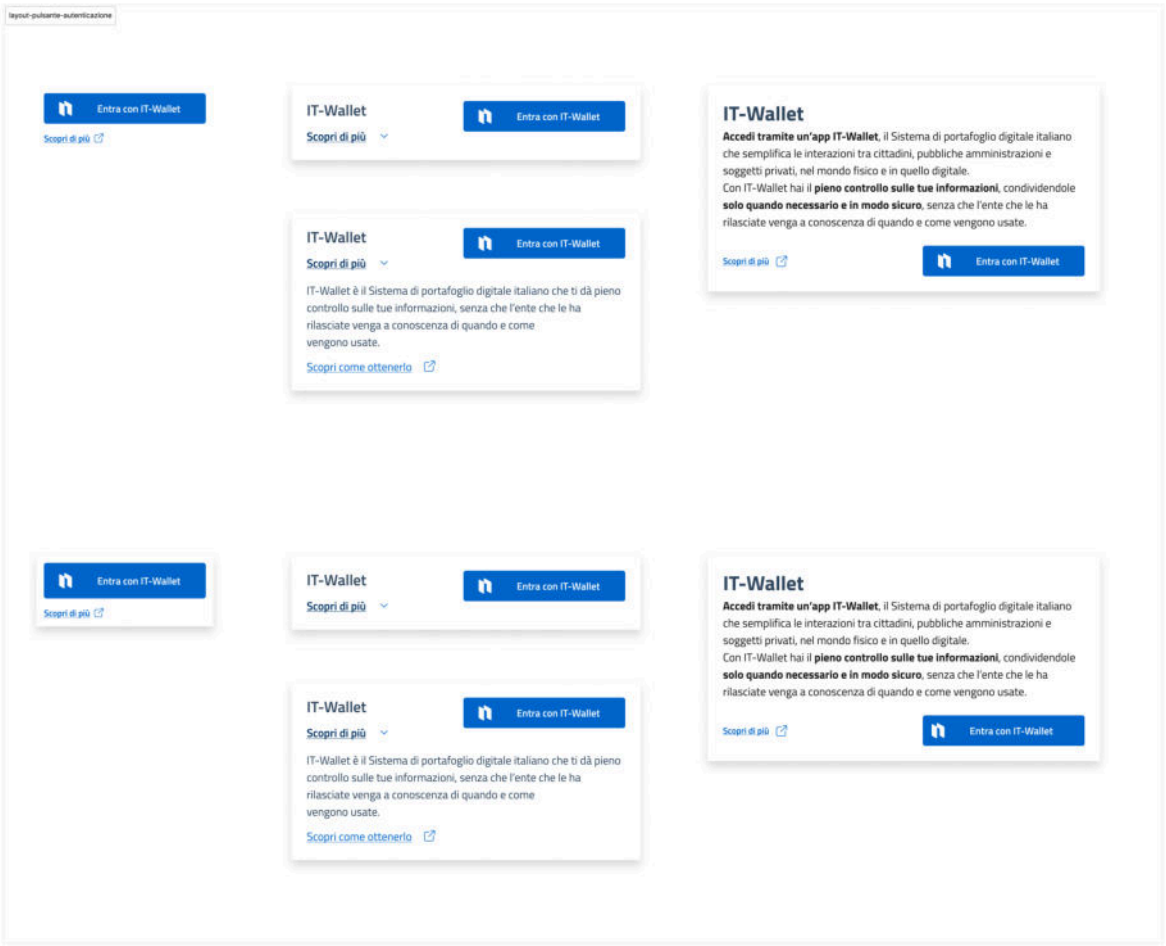
I Verificatori di Attestati Elettronici DEVONO rendere disponibile l'Authentication Button all'interno della Discovery Page delle proprie Soluzioni Tecniche per permettere all'Utente di Autenticarsi ai propri servizi tramite un'Istanza del Wallet.

L'Authentication Button è caratterizzato dai seguenti requisiti:

- DEVE essere implementato utilizzando esclusivamente quanto reso disponibile nelle official-resources:Risorse Ufficiali, e NON DEVE essere ricostruito ad hoc;
- DEVE essere utilizzato esclusivamente nelle forme, colori e proporzioni stabilite e NON DEVE essere alterato, distorto o nascosto;
- DEVE adattarsi a tutte le risoluzioni di schermo e DEVE essere integrato nella Discovery Page in modo da garantirne i requisiti minimi di usabilità e accessibilità;
- DEVE mantenere una distanza minima da altri elementi (`quiet zone`) di almeno 24px;
- DEVE riportare la dicitura "Entra con IT-Wallet";
- DOVREBBE essere sempre accompagnato da un link esterno (ad es. "Scopri di più") che rimanda al sito ufficiale del Sistema IT-Wallet, indicato nella sezione official-resources: Risorse Ufficiali;
- Gli attori che intendono integrare l'Authentication Button nella propria Soluzione Tecnica DEVONO garantirne la traduzione in altre lingue, almeno quella inglese;
- Qualora lo spazio a disposizione lo consenta e/o il contesto lo richieda, l'Authentication Button DOVREBBE essere accompagnato da un testo descrittivo: "IT-Wallet è il sistema italiano di portafogli digitali che ti permette di autenticarti online e di accedere a servizi pubblici e privati in modo sicuro e veloce, utilizzando i dati, i documenti e le qualifiche che puoi ottenere in digitale in uno dei portafogli che ne fanno parte.".

Di seguito alcuni esempi non normativi di layout dell'Authentication Button:

Varianti di Authentication Button


layout-pulsante-autenticazione
Entra con IT-Wallet
Scopri di più
IT-Wallet
Scopri di più
Entra con IT-Wallet
IT-Wallet
Scopri di più
Entra con IT-Wallet
IT-Wallet
Accedi tramite un'app IT-Wallet, il Sistema di portafoglio digitale italiano che semplifica le interazioni tra cittadini, pubbliche amministrazioni e soggetti privati, nel mondo fisico e in quello digitale.
Con IT-Wallet hai il pieno controllo sulle tue informazioni, condividendole solo quando necessario e in modo sicuro, senza che l'ente che le ha rilasciate venga a conoscenza di quando e come vengono usate.
Scopri di più
Entra con IT-Wallet
IT-Wallet è il Sistema di portafoglio digitale italiano che ti dà pieno controllo sulle tue informazioni, senza che l'ente che le ha rilasciate venga a conoscenza di quando e come vengono usate.
Scopri come ottenerlo
Entra con IT-Wallet
Scopri di più
IT-Wallet
Scopri di più
Entra con IT-Wallet
IT-Wallet
Accedi tramite un'app IT-Wallet, il Sistema di portafoglio digitale italiano che semplifica le interazioni tra cittadini, pubbliche amministrazioni e soggetti privati, nel mondo fisico e in quello digitale.
Con IT-Wallet hai il pieno controllo sulle tue informazioni, condividendole solo quando necessario e in modo sicuro, senza che l'ente che le ha rilasciate venga a conoscenza di quando e come vengono usate.
Scopri di più
Entra con IT-Wallet
IT-Wallet
Scopri di più
Entra con IT-Wallet
IT-Wallet è il Sistema di portafoglio digitale italiano che ti dà pieno controllo sulle tue informazioni, senza che l'ente che le ha rilasciate venga a conoscenza di quando e come vengono usate.
Scopri come ottenerlo

Le modalità di integrazione dell'Authentication Button nella Discovery Page possono essere molteplici a seconda del layout della pagina stessa. Di seguito alcuni esempi illustrativi e non esaustivi di Discovery Page, rispettivamente con struttura a griglia, a tab e in lista.

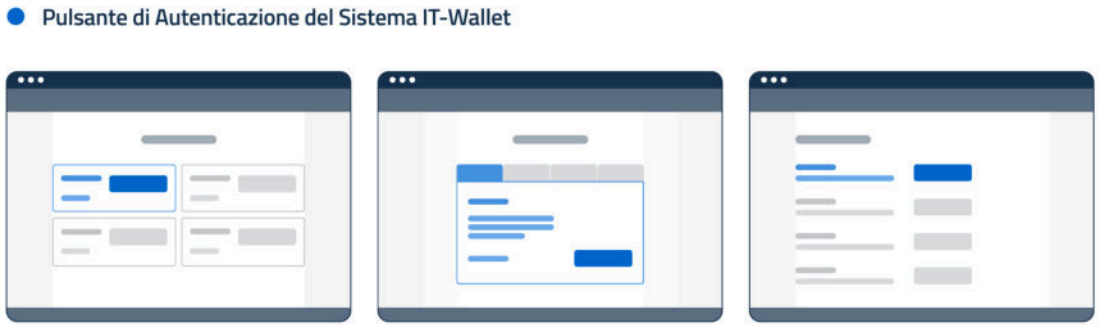


Fig. 5.39: Esempi di layout di Discovery Page a griglia, a tab e in lista

Per maggiori dettagli sull'utilizzo dell'Authentication Button vedi la sezione *Autenticazione* (pagina 80).

Authentication Button "Entra con IT-Wallet" - codice html

Il pulsante è disponibile in tre varianti di dimensione (S - default / M / L) come da Design System .Italia, ed in formato "get" (chiamata ad una pagina esterna) e "post" (form interna al pulsante). Oltre alle varianti di dimensione, sono previste due varianti di pulsante a larghezza fissa utilizzabili in situazioni in cui è preferibile mantenere coerenza con altri pulsanti simili:

- la versione giustificata (con icona a sinistra e testo centrato);
- la versione centrata (con icona e testo centrati).

La Risorsa Ufficiale dell'Authentication button html è disponibile nella sezione official-resources: Risorse Ufficiali di queste Specifiche Tecniche.

Authentication Button "Entra con IT-Wallet" - svg Per approfondimenti sull'Authentication button consultare il Brand Manual, indicato nella sezione official-resources:Risorse Ufficiali. La Risorsa Ufficiale dell'Authentication button è disponibile nella sezione official-resources: Risorse Ufficiali di queste Specifiche Tecniche.



Fig. 5.40: Authentication button in nelle varianti di dimensione (S, M, L)



Fig. 5.41: Authentication button giustificato, a larghezza fissa



Fig. 5.42: Authentication button centrato, a larghezza fissa

5.6 Gestione degli Attestati Elettronici

Il Fornitore di Wallet, attraverso la propria Soluzione Wallet, e il PID Provider o il Fornitore di Attestati Elettronici di Attributi, attraverso Touchpoint dedicati, DEVONO dare all'Utente la possibilità di gestire in ogni momento i propri Attestati Elettronici.

In questa sezione sono illustrate tre diverse categorie di requisiti per la gestione del ciclo di vita di ogni Attestato Elettronico e nello specifico per la gestione:

- **del suo stato**: per consentire all'Utente di appurare la condizione di validità o invalidità di un Attestato Elettronico;
- **dei suoi utilizzi**: per consentire all'Utente di visualizzare e gestire lo storico delle transazioni effettuate utilizzando un Attestato Elettronico;
- **dei suoi dati**: per consentire all'Utente di archiviare e ripristinare ogni Attestato Elettronico di Attributi in linea col principio di *data portability*.

Di seguito i principali aspetti che impattano e determinano l'Esperienza Utente nell'ambito della gestione degli Attestati Elettronici per mezzo di un'Istanza del Wallet e i requisiti funzionali riferiti a ciascuna categoria.

5.6.1 Stato degli Attestati Elettronici

Per garantire l'affidabilità e promuovere il corretto utilizzo della propria Soluzione Wallet, il Fornitore di Wallet DEVE dare all'Utente visibilità dello stato degli Attestati Elettronici ottenuti nella propria Istanza del Wallet sulla base delle informazioni ricevute dal Fornitore di Attestati Elettronici che ne gestisce il ciclo di vita.

Gli stati ammissibili per un Attestato Elettronico sono i seguenti:

- **Valido** (`valid`): l'EAA risulta emesso senza alcuna evidenza di criticità o problematiche. In questo caso, l'Istanza del Wallet DEVE garantire all'Utente piene funzionalità a livello di presentazione, sia in prossimità che in remoto;
- **Sospeso** (`suspended`): l'EAA risulta temporaneamente non valido, in una condizione di reversibilità (es. patente di guida sospesa). In questo caso, l'Istanza del Wallet:
  - DEVE darne all'Utente adeguata evidenza;
  - DEVE invitare l'Utente ad attendere che l'EAA torni automaticamente in stato valido alla fine del periodo di sospensione;
  - DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.

- **Da aggiornare** (`update` o `attribute_update`): l'EAA risulta non valido in quanto una o più delle sue informazioni, a livello di metadata o di Attributi, è obsoleta. In questo caso, l'Istanza del Wallet:
  - Qualora ad essere obsoleto sia uno o più dei metadata dell'EAA:
    * DOVREBBE recepire in automatico l'aggiornamento dell'EAA gestito dal Fornitore di Attestati Elettronici senza informare e richiedere azioni da parte dell'Utente.
  - Qualora ad essere obsoleto sia uno o più degli Attributi dell'EAA:
    * DEVE darne all'Utente adeguata evidenza;
    * DEVE invitare l'Utente ad aggiornare l'EAA attraverso un nuovo processo di emissione;
    * DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.
- **Non valido** (`invalid`): l'EAA risulta non valido, in una condizione di irreversibilità (es. patente di guida revocata). In questo caso, l'Istanza del Wallet:
  - DEVE darne all'Utente adeguata evidenza;
  - DEVE invitare l'Utente ad aggiornare o eliminare l'EAA a seconda che sia o meno titolato a riottenerlo e, quindi, ad usarlo ancora (es. patente di guida rinnovata – e quindi riottenibile sotto forma di EAA - oppure prescrizione medica utilizzata – e quindi non più riottenibile sotto forma di EAA);
  - DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.

Oltre agli stati sopra elencati, è bene specificare che un EAA è soggetto a scadenza. Si distinguono due tipologie di scadenza:

- **La scadenza amministrativa**: caratterizza alcuni EAA ed è fornita dalla Fonte Autentica tra gli Attributi dell'EAA stesso (es. data di scadenza della patente di guida);
- **La scadenza tecnica**: caratterizza tutti gli EAA ed è definita dalla Fonte Autentica in sinergia con il Fornitore di Attestati Elettronici con l'obiettivo di mitigare i rischi di sicurezza. Tale scadenza è generalmente fissata a 1 anno o comunque a un periodo inferiore o uguale alla data di scadenza amministrativa.

Di conseguenza, l'EAA assume i seguenti ulteriori stati:

- **In scadenza**: l'EAA risulta valido ma vicino alla data di scadenza amministrativa, se resa disponibile dalla Fonte Autentica, oppure alla data di scadenza tecnica. In questo caso, l'Istanza del Wallet:
  - Qualora ad essere vicina sia la data di scadenza amministrativa (es. mancano 30 giorni):
    * DOVREBBE darne all'Utente adeguata evidenza;
    * DOVREBBE invitare l'Utente a compiere eventuali azioni necessarie per poter riottenere l'EAA aggiornato (es. rinnovare la patente di guida presso gli uffici competenti);
    * DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto.
  - Qualora ad essere vicina sia la data di scadenza tecnica (es. mancano 7 giorni):
    * DOVREBBE innescare in automatico l'aggiornamento dell'EAA gestito dal Fornitore di Attestati Elettronici senza richiedere azioni da parte dell'Utente.
    * Se l'aggiornamento automatico va a buon fine e restituisce l'EAA aggiornato:
    * DEVE sostituirlo al precedente senza generare duplicazioni inattese, disservizi o notifiche all'Utente;
    * Se l'aggiornamento automatico va a buon fine ma non restituisce alcun EAA:
    * DEVE adeguatamente informare l'Utente circa la potenziale perdita di titolarità nei confronti dell'EAA stesso;
    * DEVE invitare l'Utente ad aggiornare o eliminare l'EAA a seconda che sia o meno titolato a riottenerlo e, quindi, ad usarlo ancora (es. patente di guida rinnovata – e quindi riottenibile sotto forma di EAA - oppure prescrizione medica utilizzata – e quindi non più riottenibile sotto forma di EAA);
    * DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.
    * Se l'aggiornamento automatico non dovesse andare a buon fine per indisponibilità del servizio o altri errori tecnici:
    * DOVREBBE effettuare almeno un nuovo tentativo di aggiornamento prima che l'EAA superi la data di scadenza tecnica.
- **Scaduto**: l'EAA risulta valido ma ha superato la data di scadenza amministrativa, se resa disponibile dalla Fonte Autentica, oppure ha superato la data di scadenza tecnica. In questo caso, l'Istanza del Wallet:

  - Qualora sia stata superata la data di scadenza amministrativa:
    * DEVE darne all'Utente adeguata evidenza;
    * DEVE invitare l'Utente a compiere eventuali azioni necessarie per riottenere l'EAA aggiornato (es. rinnovare la patente presso gli uffici competenti);
    * DEVE garantire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.
  - Qualora sia stata superata la data di scadenza tecnica (aggiornamento automatico non effettuato o fallito):
    * DEVE darne all'Utente adeguata evidenza, distinguendo il caso specifico da quello di un'eventuale scadenza amministrativa;
    * DEVE invitare l'Utente ad aggiornare l'EAA, ossia ad intraprendere un flusso di riemissione dell'EAA stesso;
    * DEVE consentire all'Utente la funzionalità di presentazione, sia in prossimità che in remoto. Conseguentemente, l'Istanza di Relying Party DEVE adeguatamente informare il Verificatore circa lo specifico stato dell'EAA.

Di seguito i requisiti funzionali a supporto dell'Esperienza Utente relativi all'aggiornamento dell'Attestato Elettronico che il Fornitore di Attestati Elettronici DEVE garantire attraverso la Soluzione Wallet:

- L'Utente visualizza nella Vista in Anteprima dell'Attestato che il suo stato è diverso da valido;
- L'Utente visualizza nella Vista di Dettaglio un messaggio che lo informa del nuovo stato dell'Attestato e PUÒ scoprire maggiori informazioni;
- L'Utente visualizza eventuali informazioni aggiuntive sui requisiti e/o limitazioni relative allo stato dell'Attestato Elettronico di Attributi e PUÒ chiudere il messaggio oppure procedere con un eventuale azione richiesta dal Fornitore di Attestati Elettronici.
- L'Utente visualizza all'interno della Vista di Dettaglio apposite Call To Action per eliminare l'Attestato Elettronico o aggiornarlo se in stato non più valido.

Il flusso è rappresentato di seguito con wireframe esemplificativi.

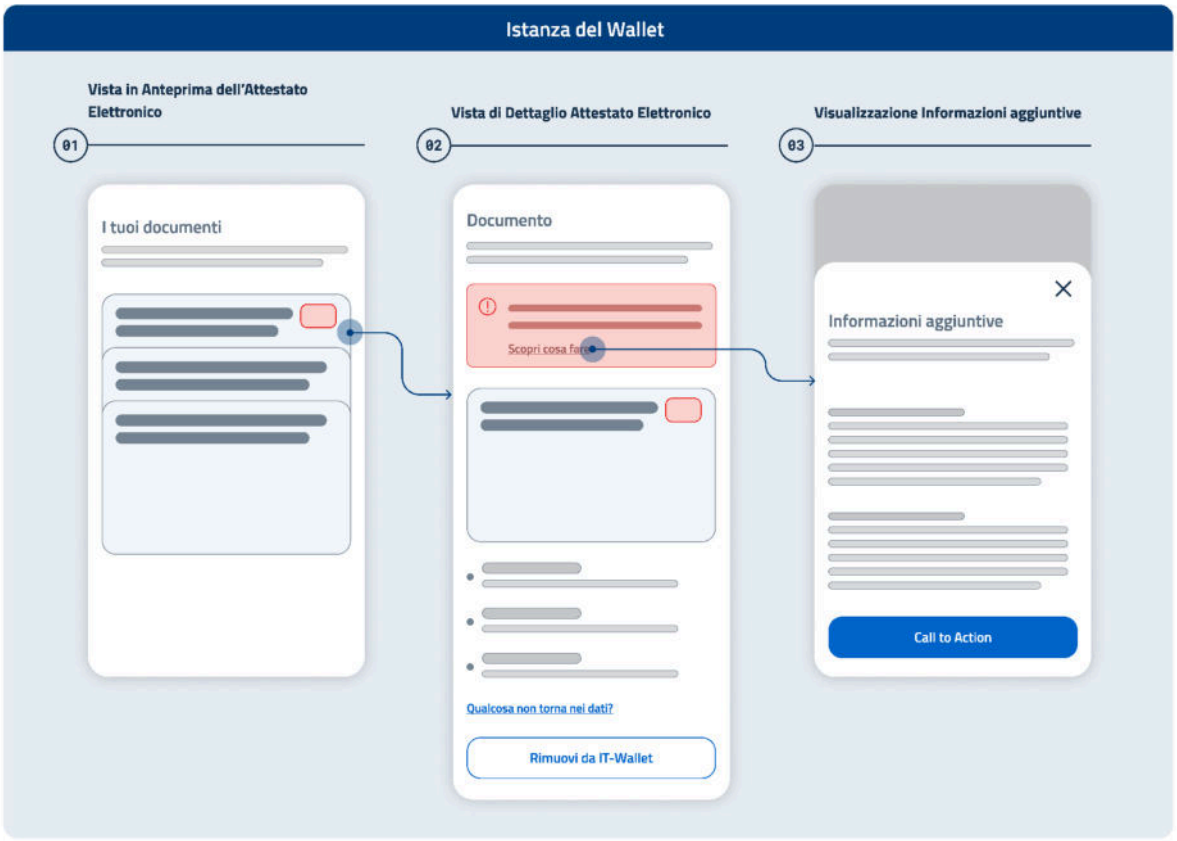


Fig. 5.43: Esempio di Esperienza Utente nell'Aggiornamento di un Attestato Elettronico

Revoca degli Attestati Elettronici

La revoca si configura come quel meccanismo che determina il passaggio di un Attestato Elettronico da uno stato valido a uno stato non valido. La revoca può avvenire in modalità attiva o passiva:

- **Revoca attiva**: ovvero la revoca di un Attestato Elettronico su richiesta dell'Utente. Tale processo comporta la sola revoca dell'Attestato Elettronico e non del corrispettivo documento fisico, se esistente. Di seguito un elenco esemplificativo di scenari in cui il Fornitore di Wallet DEVE dare possibilità all'Utente di richiedere la revoca di un Attestato Elettronico:
  - l'Utente decide di non voler più possedere uno specifico Attestato Elettronico;
  - l'Utente decide di disattivare la propria Istanza del Wallet e, di conseguenza, tutti gli Attestati Elettronici precedentemente ottenuti;
  - l'Utente non è più in possesso del dispositivo su cui è installata la propria Istanza del Wallet a causa di uno smarrimento o un furto.
- **Revoca passiva**: ovvero la revoca di un Attestato Elettronico gestita dal relativo Fornitore di Attestati Elettronici per conto della Fonte Autentica. In questo caso l'Istanza del Wallet DEVE informare l'Utente del cambiamento di stato dell'Attestato Elettronico e, in aggiunta, il Fornitore di Attestati Elettronici PUÒ informare l'Utente attraverso altri Touchpoint. Di seguito un elenco esemplificativo di scenari che porterebbero alla revoca di un Attestato Elettronico:
  - il documento fisico corrispondente all'Attestato Elettronico è stato dichiarato smarrito o danneggiato da parte dell'Utente tramite apposito canale/ Touchpoint;
  - il documento fisico corrispondente all'Attestato Elettronico è stato revocato dalle autorità competenti;
  - non sussistono più i requisiti minimi di sicurezza e/o affidabilità di una o più delle parti coinvolte;
  - il dispositivo dell'Utente non soddisfa più i requisiti minimi di sicurezza (dispositivo rootato o jailbroken).

Di seguito i requisiti di Esperienza Utente che il Wallet Provider DEVE garantire attraverso la propria Soluzione Wallet: - l'Utente apre la Vista di Dettaglio dell'Attestato Elettronico che vuole revocare; - l'Utente seleziona la Call to Action per la revoca dell'Attestato Elettronico; - l'Utente prende visione di tutte le informazioni rilevanti l'azione in corso e dà il proprio consenso per proseguire oppure lo nega per annullare l'operazione; - l'Utente visualizza l'esito positivo della revoca avvenuta.

Il flusso è rappresentato di seguito con wireframe esemplificativi.



Fig. 5.44: Esempio di Esperienza Utente nella Revoca di un Attestato Elettronico da Wallet

5.6.2 Registrazione delle Transazioni

Al fine di garantire i principi di visibilità e trasparenza, il Fornitore di Wallet DEVE mettere a disposizione, nell'Istanza del Wallet, una dashboard intuitiva che consenta all'Utente di visualizzare lo storico delle transazioni effettuate tramite la propria Istanza del Wallet (ad esempio, emissione o presentazione di Attestati Elettronici). In particolare, la dashboard DEVE:

- fornire una panoramica di tutte le transazioni registrate e consentire all'Utente di accedere a viste di dettaglio delle singole transazioni;
- consentire all'Utente, qualora una transazione coinvolga una Relying Party, di avviare facilmente una richiesta di cancellazione dei dati verso la relativa Relying Party (utilizzando le informazioni di contatto registrate);
- supportare l'esportazione di uno o più record di transazione in un file;
- consentire all'Utente di cancellare uno o più record di transazione, previa adeguata informativa.

5.6.3 Archiviazione e ripristino degli Attestati Elettronici di Attributi

Con l'obiettivo di garantire il principio di data portability, la Soluzione Wallet DEVE garantire all'Utente l'accesso a specifiche funzionalità per:

- richiedere l'archiviazione, quindi il salvataggio, degli Attestati Elettronici di Attributi ottenuti su una specifica Istanza del Wallet;
- richiedere il ripristino dei propri Attestati Elettronici di Attributi su un'altra Istanza del Wallet.

5.7 Disattivazione dell'Istanza del Wallet

La disattivazione dell'Istanza del Wallet è la funzionalità che rende l'Istanza del Wallet disattiva e quindi non più operativa. La disattivazione dell'Istanza del Wallet può essere scatenata da attori differenti a seconda delle circostanze, in particolare:

- da parte dell'Utente nel caso in cui, ad esempio:
  - il dispositivo sia stato smarrito o rubato;
  - il dispositivo risulti compromesso;
  - il dispositivo sia stato resettato alle impostazioni di fabbrica.
- da parte di un ente terzo titolato nel caso in cui, ad esempio:
  - la Soluzione Wallet non rispetti più i requisiti minimi di sicurezza.

Il Fornitore di Wallet DEVE, quindi, garantire all'Utente la possibilità di disattivare volontariamente la propria Istanza del Wallet tramite:

- l'Istanza del Wallet stessa;
- un Touchpoint (e.g. un sito web) fornito dal Fornitore di Wallet;
- l'app store del proprio dispositivo, disinstallando l'Istanza del Wallet.

Di seguito i requisiti funzionali a supporto dell'Esperienza Utente relativi alla disattivazione dell'Istanza del Wallet che il Fornitore di Wallet DEVE garantire attraverso la propria Soluzione Wallet:

- l'Utente accede alla propria Istanza del Wallet utilizzando la modalità di sblocco precedentemente impostata oppure si Autentica presso il Touchpoint reso disponibile dal Fornitore di Wallet;
- l'Utente seleziona la funzionalità di disattivazione dell'Istanza del Wallet;
- l'Utente viene informato che la disattivazione dell'Istanza del Wallet renderà non più utilizzabili gli Attestati Elettronici precedentemente ottenuti;

- l'Utente conferma l'azione per procedere con la disattivazione oppure annulla l'operazione;
- l'Utente visualizza l'esito positivo della disattivazione avvenuta;
- l'Utente, in caso di nuovo accesso, prende visione del fatto che l'Istanza del Wallet è disattiva.

L'Utente ha la possibilità di riattivare l'Istanza del Wallet riscaricando l'app dall'app store, in caso di cancellazione, e/o ripercorrendo il processo di attivazione. Per approfondimenti si rimanda alla sezione *Attivazione dell'Istanza del Wallet* (pagina 41).

Una volta riattivata l'Istanza del Wallet, gli Attestati Elettronici di Attributi potranno essere ri-ottenuti avviando nuovamente il processo di ottenimento o di ripristino. Per approfondimenti si rimanda rispettivamente alle sezioni *Ottenimento degli Attestati Elettronici di Attributi* (pagina 46) e *Archiviazione e ripristino degli Attestati Elettronici di Attributi* (pagina 107).

In caso di errori nell'utilizzo dell'Istanza del Wallet, il Fornitore di Wallet DEVE garantire all'Utente la visualizzazione di messaggi coerenti che lo informino e guidino alla loro risoluzione. Per approfondimenti si rimanda alla sezione *Gestione degli errori* (pagina 109).

Il flusso è rappresentato di seguito con wireframe esemplificativi.



Fig. 5.45: Esempio di Esperienza Utente nella Disattivazione di un'Istanza del Wallet

5.8 Gestione degli errori

Il Sistema IT-Wallet prevede l'interazione di una molteplicità di servizi erogati da diversi attori. È quindi importante che venga definito un modello di gestione degli errori efficace, con l'obiettivo di migliorare la percezione e il senso di affidabilità dell'interno ecosistema e permettere all'Utente di sentirsi guidato nell'interazione con le diverse Soluzioni Tecniche e nella gestione consapevole di eventuali criticità durante la fruizione del servizio.

Una comunicazione efficace in caso di errore determina un vantaggio anche per gli attori coinvolti, in quanto concorre alla riduzione delle richieste di assistenza e, quindi, alla minimizzazione dell'impatto sui sistemi.

Ciascun Attore Primario DEVE implementare una corretta gestione degli errori, in conformità alle attuali Specifiche Tecniche, al fine di comunicarli, direttamente o indirettamente, all'Utente e tramite l'Istanza del Wallet. Gli errori possono essere declinati, sulla base della loro natura, come segue:

- **la fase dell'Esperienza Utente** in cui l'errore può verificarsi: attivazione o disattivazione dell'Istanza del Wallet, ottenimento, presentazione o gestione degli Attestati Elettronici;
- **la tipologia di errore**: di sistema, di comunicazione tra attori, etc.;
- **l'attore responsabile dell'errore**: Fornitore di Wallet, PID Provider, Fornitore di Attestati Elettronici di Attributi, Fonte Autentica;
- **la modalità di restituzione dell'errore**: messaggio in pagina, banner, toast message, etc.;
- **le azioni suggerite all'Utente per risolvere l'errore**: suggerimento di attesa, richiesta di effettuare un nuovo tentativo, rimando alle domande frequenti e/o al servizio di assistenza, etc.;
- **le modalità di presa in carico dell'errore**: apertura di una richiesta di assistenza tramite l'Istanza del Wallet, rimando ad altri canali di approfondimento, etc. Per approfondimenti si rimanda alla sezione *Assistenza Utente* (pagina 112).

Di seguito, una lista non esaustiva delle principali casistiche di errore, con riferimento all'attore responsabile della loro gestione, per ciascuna fase dell'Esperienza Utente. La lista dettagliata degli errori da gestire per ogni endpoint di interazione con l'Utente è disponibile nelle sottosezioni dedicate agli errori all'interno di *Endpoints* (pagina 571).

5.8.1 Errori di Attivazione dell'Istanza del Wallet

Tipologia di errore	Attore responsabile
Il dispositivo non supporta la Soluzione Wallet (e.g. assenza dei requisiti minimi di sicurezza o tecnologici)	Fornitore di Wallet
I servizi del Fornitore di Wallet non rispondono (e.g. errori tecnici o assenza connessione)	Fornitore di Wallet
I servizi del PID Provider o del fornitore dell'IT-Wallet ID non rispondono (e.g. errori tecnici)	PID Provider o fornitore dell'IT-Wallet ID
Il processo di Autenticazione sul servizio del Gestore di Identità Digitale non è andato a buon fine (e.g. errori tecnici, identità non riconosciuta, etc.)	Gestore di Identità Digitale

Nota

Quando la verifica del documento elettronico viene eseguita in aggiunta all'autenticazione del Gestore di Identità Digitale, possono verificarsi scenari di errore aggiuntivi. Per codici di errore dettagliati e procedure di gestione, vedere *Gestione Errori* (pagina 499).

5.8.2 Errori di ottenimento degli Attestati Elettronici di Attributi

Tipologia di errore	Attore responsabile
L'Istanza del Wallet e/o il PID / IT-Wallet ID non risultano validi	Fornitore di Wallet
Il servizio di ottenimento di un Attestato Elettronico di Attributi non è disponibile (e.g. errori tecnici)	Fornitore di Attestati Elettronici di Attributi, Fonte Autentica
L'Utente non riesce ad ottenere nella propria Istanza del Wallet un certo Attestato Elettronico di Attributi (e.g. assenza di titolarità, versione fisica non valida o scaduta, etc.)	Fonte Autentica
Il servizio di ottenimento di un Attestato Elettronico di Attributi non può essere elaborato entro l'intervallo di tempo definito	Fornitore di Attestati Elettronici di Attributi
Il servizio di ottenimento di un Attestato Elettronico di Attributi non può essere completato in modo sincrono. All'Utente è richiesto di attendere finché l'Attestato non diventa disponibile (flusso differito)	Fornitore di Attestati Elettronici di Attributi, Fonte Autentica
Il servizio di ottenimento di un Attestato Elettronico di Attributi non può essere elaborato poiché la richiesta supera il limite consentito (e.g. più Attestati richiesti contemporaneamente)	Fornitore di Attestati Elettronici di Attributi

5.8.3 Errori di presentazione degli Attestati Elettronici

Tipologia di errore	Attore responsabile
L'Utente non possiede all'interno della propria Istanza del Wallet gli Attributi contenuti in uno o più Attestati Elettronici richiesti per la fruizione di un determinato servizio	Fornitore di Wallet
I servizi del Fornitore di Wallet e/o del Verificatore di Attestati Elettronici non rispondono (e.g. errori tecnici o assenza connessione)	Fornitore di Wallet, Verificatore di Attestati Elettronici

5.8.4 Errori di gestione degli Attestati Elettronici

Tipologia di errore	Attore responsabile
Il servizio di revoca / archiviazione/ ripristino di un Attestato Elettronico di Attributi non è disponibile (e.g. errori tecnici)	Fornitore di Attestati Elettronici di Attributi
Il servizio di revoca del PID non è disponibile (e.g. errori tecnici)	PID Provider
Il servizio di cancellazione delle informazioni inviate a una Relying Party non è disponibile (e.g. errori tecnici)	Relying Party

5.8.5 Errori di disattivazione dell'Istanza del Wallet

Tipologia di errore	Attore responsabile
Il servizio di disattivazione dell'Istanza del Wallet non è disponibile (e.g. errori tecnici)	Fornitore di Wallet

Oltre alla gestione degli errori, DEVE essere garantita da parte di tutti gli Attori Primari anche la gestione di esiti negativi dovuti alla volontà dell'Utente di abbandonare o annullare un flusso (es. attivazione, ottenimento, presentazione, etc.). In questi casi DEVE essere previsto un feedback che dia conferma all'Utente della scelta intrapresa e che PUÒ includere una Call to Action per proseguire.

5.9 Assistenza Utente

Per un'efficace gestione degli errori e di eventuali altre problematiche, gli Attori Primari DEVONO garantire un'adeguata assistenza all'Utente, strutturando un modello di assistenza semplice ed efficace basato sui seguenti principi:

- **Risoluzione autonoma**: permettere all'Utente di consultare domande frequenti (FAQ) sui contenuti e sulle funzionalità dell'Istanza del Wallet, al fine di risolvere eventuali casistiche di errore o problematiche in maniera autonoma.
- **Apertura guidata di una segnalazione**: guidare l'Utente all'eventuale apertura di una segnalazione, in modo da circoscrivere la problematica e facilitare la sua gestione.
- **Collaborazione tra attori**: rendere possibile un adeguato coordinamento tra tutti gli attori coinvolti (Fornitore di Wallet, Fornitore di Attestati Elettronici di Attributi, PID Provider e Fonte Autentica) in base al proprio ruolo e alle modalità operative specifiche.

- **Comunicazione efficiente**: garantire all'Utente la possibilità di monitorare lo stato aggiornato della propria richiesta durante tutte le fasi di lavorazione, attraverso una comunicazione chiara, continua e coordinata.

Per applicare queste buone pratiche, gli attori coinvolti DOVREBBERO implementare i seguenti livelli di assistenza gerarchici:

1. **I Livello – Gestione autonoma**: il Fornitore di Wallet DOVREBBE permettere all'Utente di disporre di una sezione di domande frequenti (FAQ) all'interno della propria Istanza del Wallet per chiarire dubbi e risolvere in autonomia alcune problematiche. Ogni attore DOVREBBE formulare delle domande frequenti e relative risposte specifiche rispetto a dati e funzionalità messe a disposizione al Fornitore di Wallet o nei propri Touchpoint. Per alcune casistiche di errore, il Fornitore di Wallet DOVREBBE rendere direttamente disponibile il canale di assistenza di un altro attore, per facilitare una gestione tempestiva ed evitare l'apertura di una richiesta di assistenza nell'Istanza del Wallet.
2. **II Livello – Richiesta di assistenza al Fornitore di Wallet**: se il I livello non fosse sufficiente, il Fornitore di Wallet DOVREBBE permettere all'Utente di aprire una o più richieste di assistenza, effettuare una diagnosi e procedere alla risoluzione della problematica, se di sua competenza. Tali richieste DOVREBBERO essere gestite tramite l'Istanza del Wallet o altri Touchpoint del Fornitore di Wallet.
3. **III Livello – Inoltro della richiesta all'attore responsabile della problematica**: se il II livello non fosse sufficiente, il Fornitore di Wallet DOVREBBE garantire che la richiesta sia inoltrata all'attore responsabile (Fornitore di Attestati Elettronici di Attributi, PID Provider e Fonte Autentica) che si fa carico della risoluzione della problematica e comunicare l'esito all'Utente.

Di seguito i requisiti di Esperienza Utente che il Fornitore di Wallet DEVE garantire attraverso la propria Soluzione Wallet:

- l'Utente ha accesso a modalità di assistenza in ogni momento dell'Esperienza Utente, attraverso una chiara indicazione del punto di accesso;
- l'Utente ha la possibilità di aprire una richiesta di assistenza tramite la propria Istanza del Wallet o altri Touchpoint messi a disposizione dal Fornitore di Wallet;
- nel caso di apertura di una richiesta di assistenza, l'Utente riceve conferma tempestiva dell'avvenuta presa in carico;
- l'Utente è informato preventivamente nel caso in cui sia necessario condividere i propri dati con soggetti terzi;
- l'Utente è informato nei casi in cui la richiesta di assistenza debba essere gestita al di fuori della propria Istanza del Wallet, quindi su canali terzi;
- l'Utente monitora l'esito della richiesta in ogni momento attraverso funzionalità che DEVONO essere messe a disposizione dagli attori che hanno preso in carico la richiesta.

5.10 Feedback Utente

La raccolta dei feedback degli Utenti permette di monitorare l'Esperienza Utente, identificare le aree per una possibile ottimizzazione e misurare l'efficacia del servizio in maniera continuativa. Ogni Fornitore di Wallet DOVREBBE predisporre un sistema strutturato di raccolta feedback, per monitorare e migliorare l'Esperienza Utente.

Tale sistema di feedback POTREBBE essere alimentato da due diverse tipologie di raccolta feedback:

- **feedback transazionali** (Customer Effort Score, Customer Satisfaction): raccolta contestuale ad azioni specifiche, come l'aggiunta di un Attestato Elettronico o il completamento di un'operazione di presentazione e verifica;
- **feedback relazionali** (Net Promoter Score): raccolta non contestuale ad azioni specifiche, per la misurazione della percezione generale dell'Utente, in termini di soddisfazione, fedeltà e possibile raccomandazione ad Utenti terzi.

Di seguito, le indicazioni proposte per l'implementazione di tali tipologie di feedback:

Raccolta di feedback transazionale

- **Customer Effort Score (CES)**: per misurare la facilità di utilizzo delle funzionalità POSSONO essere predisposti questionari, ad esempio tramite componenti quali modali o pop-up nell'Istanza del Wallet, al termine di azioni specifiche o specifici processi, ad esempio:
  - a conclusione del processo di ottenimento di un Attestato Elettronico;
  - a conclusione del processo di Autenticazione, se positivo;
  - a conclusione del processo di presentazione, in particolare a conclusione della prima occasione di presentazione e non più di una volta ogni 6 mesi;
  - a conclusione dei processi di revoca e disattivazione, per approfondirne le motivazioni.
- **Customer Satisfaction Survey (CSAT)**: per misurare la soddisfazione generale dell'Utente dopo un periodo prolungato di utilizzo dell'Istanza del Wallet POSSONO essere predisposti questionari, ad esempio tramite componenti quali modali o pop-up nell'Istanza del Wallet. Si consiglia di utilizzare il CSAT ad intervalli non inferiori a sei mesi e come alternativa al CES, per evitare di somministrare questionari con troppa frequenza.

Raccolta di feedback relazionale

- **Net Promoter Score (NPS)**: per misurare la fedeltà dell'Utente e la probabilità di raccomandazione ad Utenti terzi, si PUÒ richiedere di esprimere una valutazione una o due volte l'anno attraverso lo stesso canale di erogazione del servizio (e.g. l'Istanza del

Wallet) o canali esterni, quali e-mail o SMS, e comunque in linea con la strategia di raccolta feedback adottata.

CAPITOLO 6

L'Infrastruttura di Trust

L'ecosistema IT-Wallet opera all'interno di un'infrastruttura di trust federata dove le entità partecipanti stabiliscono relazioni di trust crittografiche e mantengono la conformità con standard di sicurezza comuni. Questa infrastruttura fornisce le fondamenta per operazioni sicure di Credenziali Elettroniche tra i partecipanti dell'ecosistema.

Questa sezione definisce l'implementazione del Trust Model in un'infrastruttura che è conforme a OpenID Federation 1.0 OID-FED. Tale infrastruttura nazionale prevede un'API RESTful per la distribuzione di metadati, policy dei metadati, trust mark, chiavi pubbliche crittografiche e certificati X.509, e lo stato di revoca dei partecipanti, chiamati anche Entità di Federazione.

Questa infrastruttura di trust lavora in coordinamento con l'Infrastruttura del Registro (vedi *Infrastruttura del Registro* (pagina 159)) per abilitare i processi di onboarding delle entità dettagliati in *Onboarding delle Entità* (pagina 239). In particolare, abilita l'implementazione tecnica dei processi di onboarding descritti in *Onboarding delle Entità* (pagina 239) e supporta gli scenari operativi illustrati in *Onboarding Journey Maps* (pagina 32).

Abilitazione dell'Onboarding: L'Infrastruttura di Trust fornisce i meccanismi crittografici che consentono alle nuove entità (Credential Issuer, Relying Party, Fornitori di Wallet) di stabilire relazioni di trust verificabili durante il loro processo di registrazione. Senza questa infrastruttura, le entità non sarebbero in grado di dimostrare il loro stato di conformità o le capacità operative agli altri partecipanti dell'ecosistema.

Supporto al Ciclo di Vita delle Entità: Durante tutto il ciclo di vita operativo di un'entità, l'Infrastruttura di Trust mantiene attestazioni di trust aggiornate, gestisce la rotazione delle chiavi, gestisce scenari di revoca e supporta il monitoraggio della conformità. Questo supporta

direttamente le procedure di gestione del ciclo di vita dettagliate in *Onboarding delle Entità* (pagina 239).

Integrazione con l'Infrastruttura del Registro: L'Infrastruttura di Trust implementa il componente Federation Registry dell'Infrastruttura del Registro più ampia, fornendo le fondamenta tecniche per la scoperta delle entità e la validazione del trust che sottende tutte le procedure di onboarding.

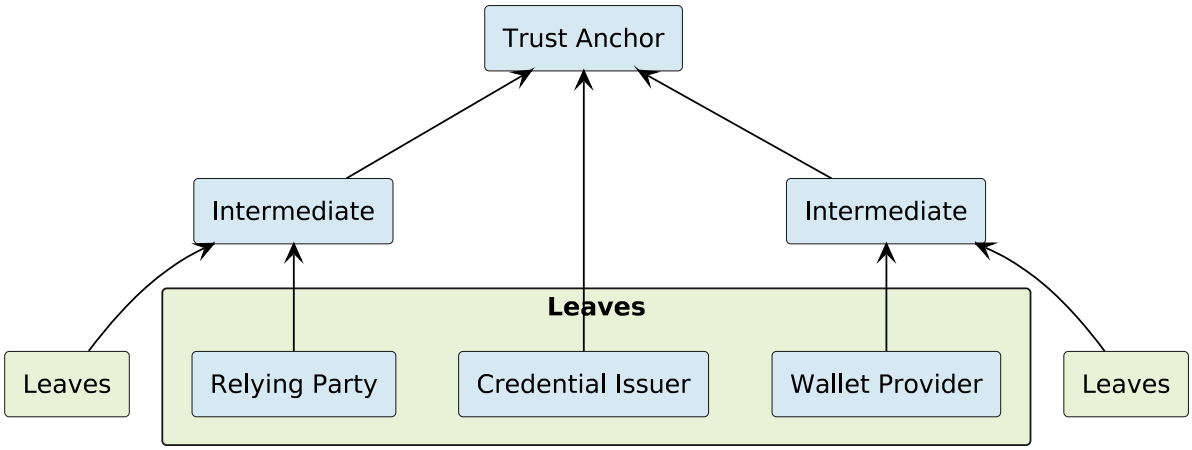


Fig. 6.1: I ruoli all'interno della Federazione, dove il National Trust Anchor supervisiona i suoi subordinati, che includono uno o più Intermediari e Foglie.

In questa rappresentazione, sia il *National Trust Anchor* che gli Intermediari assumono il ruolo di Registration Authority.

6.1 Ruoli di Federazione

Tutti i partecipanti sono Entità di Federazione che DEVONO essere registrate da un Registration Body, eccetto le Istanze del Wallet che sono dispositivi personali dell'Utente finale autenticati dal loro Fornitore di Wallet.

Nota

L'Istanza del Wallet, come dispositivo personale, è considerata affidabile attraverso un'attestazione verificabile rilasciata e firmata da una terza parte fidata.

Questo è chiamato *Wallet Instance Attestation* ed è documentato nella sezione dedicata *Emissione della Wallet Instance Attestation* (pagina 305).

Ruolo nell'Onboarding: Durante la registrazione dell'entità, il Trust Anchor e gli Intermediate agiscono come Autorità di Federazione. Ciò stabilisce la posizione del partecipante nella gerarchia di fiducia e gli consente di partecipare alle operazioni di credenziali. Le foglie (Emittenti di

credenziali, Relying Party, Wallet Provider) si sottopongono a registrazione per dimostrare la propria idoneità e ricevere l'autorizzazione a svolgere le funzioni loro assegnate.

Ruolo nelle Operazioni: Durante l'emissione e la presentazione delle Credenziali, questi ruoli abilitano la validazione del trust distribuita senza richiedere la verifica centralizzata per ogni transazione. Le Foglie utilizzano il loro stato registrato per emettere Credenziali, verificare presentazioni o fornire servizi di Wallet agli utenti finali.

Di seguito la tabella con il riepilogo dei ruoli delle Entità di Federazione, mappati sui corrispondenti ruoli EUDI Wallet.

Ruolo EUDI	Ruolo di Federazione	Note
Public Key Infrastructure (PKI)	*National Trust Anchor*	La Federazione ha capacità PKI e utilizza OpenID Federation 1.0 OID-FED. L'Entità che configura l'intera infrastruttura a livello nazionale è il *National Trust Anchor*.
Qualified Trust Service Provider (QTSP)	Leaf	
Fornitore di Attestati Elettronici di Dati di Identificazione Personale	Leaf	
Fornitore di Attestati Elettronici di Attributi Qualificati	Leaf	
Fornitore di Attestati Elettronici di Attributi	Leaf	
Relying Party	Leaf	
Trust Service Provider (TSP)	Leaf	
Fornitore di Wallet	Leaf	

6.2 Integrazione dell'Infrastruttura di Trust e del Registro

L'Infrastruttura di Trust implementa il componente Federation Registry dell'Infrastruttura del Registro. Il Federation Registry mantiene l'elenco autorevole delle entità fidate attraverso gli endpoint di federazione definiti in questa sezione, inclusi l'elenco delle entità (/list), le dichiarazioni dei subordinati (/fetch), la validazione dei trust mark (/trust_mark_status), gli eventi sui subordinati (/federation_subordinate_events_endpoint) e la gestione delle chiavi storiche (/historical-jwks).

Questo Federation Registry opera insieme ad altri componenti del registro (Claims Registry, AS Registry, Catalogo degli Attestati Elettronici, Taxonomy) per fornire supporto completo all'ecosistema. Per l'architettura completa del registro e le interazioni dei componenti, vedi *Infrastruttura del Registro* (pagina 159).

6.3 Proprietà Generali

L'architettura dell'infrastruttura di trust è fondata sui seguenti principi fondamentali:

Identificatore	Proprietà	Descrizione
P1	**Sicurezza**	Incorpora meccanismi per garantire l'integrità, la riservatezza e l'autenticità delle Relazioni di Trust e delle interazioni all'interno della federazione.
P2	**Privacy**	Progettata per rispettare e proteggere la privacy delle entità coinvolte; la divulgazione minima è parte integrante di questo principio.
P3	**Interoperabilità**	Supporta l'interazione fluida e la creazione di trust tra sistemi ed entità diverse all'interno della federazione.
P4	**Trust Transitivo**	Trust stabilita indirettamente attraverso una catena di relazioni fidate, che consente alle entità di fidarsi reciprocamente sulla base di autorità comuni e intermediari fidati.
P5	**Delega**	Capacità o funzionalità tecnica di delegare autorità o responsabilità ad altre entità, consentendo un meccanismo di trust distribuito.
P6	**Scalabilità**	Progettata per gestire in modo efficiente un numero crescente di entità o interazioni senza un aumento significativo della complessità di gestione del trust.
P7	**Flessibilità**	Adattabile a varie esigenze operative e organizzative, consentendo alle entità di definire e adeguare le proprie Relazioni di Trust e policy.
P8	**Autonomia**	Pur facendo parte di un ecosistema federato, ciascuna entità mantiene il controllo sulle proprie definizioni e configurazioni.
P9	**Decentralizzazione**	A differenza dei sistemi centralizzati tradizionali, l'infrastruttura di trust consente un approccio decentralizzato.

6.4 Requisiti dell'Infrastruttura di Trust

Questa sezione include i requisiti necessari per la corretta implementazione e il funzionamento dell'infrastruttura di trust.

Tabella 6.3: Requisiti Funzionali

ID	Descrizione
FR1	**Creazione di Trust nella Federazione**: il sistema deve essere in grado di stabilire trust tra diverse entità (Credential Issuer, Relying Party, ecc.) all'interno di una federazione, utilizzando firme crittografiche per lo scambio sicuro di informazioni sui partecipanti all'ecosistema.
FR2	**Autenticazione delle Entità**: il sistema deve implementare meccanismi per autenticare le entità all'interno della federazione, garantendone la conformità alle regole condivise.
FR3	**Validazione delle Firme**: il sistema deve supportare la creazione, verifica e validazione di firme elettroniche e fornire meccanismi standard e sicuri per ottenere le chiavi pubbliche crittografiche necessarie per la validazione delle firme.
FR4	**Marcatura Temporale**: gli artefatti firmati devono contenere marcature temporali per garantire l'integrità e il non-ripudio delle transazioni nel tempo, grazie alle interfacce, ai servizi, al modello di archiviazione e agli approcci definiti all'interno della federazione.
FR5	**Validazione dei Certificati**: il sistema richiede trasmissioni riservate, protette tramite TLS su HTTP, e la validazione dei certificati per l'autenticazione dei siti web.
FR6	**Interoperabilità e Conformità agli Standard**: garantire l'interoperabilità tra i membri della federazione aderendo agli standard tecnici, facilitando le transazioni elettroniche transfrontaliere.
FR7	**Protezione dei Dati e Privacy**: implementare misure di protezione dei dati in conformità con il GDPR, garantendo la privacy e la sicurezza dei dati personali trattati all'interno della federazione.
FR8	**Risoluzione delle Controversie e Responsabilità**: stabilire procedure chiare per la risoluzione delle controversie e definire le responsabilità tra i membri della federazione.
FR9	**Servizi di Emergenza e Revoca**: implementare meccanismi per la revoca immediata dei partecipanti in caso di violazioni della sicurezza o altre emergenze.
FR10	**Infrastruttura di Trust Scalabile**: il sistema deve supportare meccanismi scalabili per la creazione di trust, sfruttando approcci e soluzioni tecniche che complementino gli approcci di delega transitiva per gestire in modo efficiente le Relazioni di Trust man mano che la federazione cresce, eliminando i registri centrali che potrebbero presentare fallimenti tecnici o amministrativi.

continues on next page

Tabella 6.3 – continua dalla pagina precedente

ID	Descrizione
FR11	**Scalabilità Efficiente dello Storage**: implementare una soluzione di archiviazione che scala orizzontalmente per accogliere volumi di dati crescenti, minimizzando i costi di archiviazione centrale e amministrativi. Il sistema dovrebbe consentire ai membri di archiviare e presentare in modo indipendente attestazioni di trust storiche e artefatti firmati durante le risoluzioni delle controversie, mentre l'infrastruttura della federazione mantiene solo un registro delle chiavi storiche per validare i dati storici, archiviati e forniti dai partecipanti.
FR12	**Attestazione Verificabile (Trust Mark)**: incorporare un meccanismo per l'emissione e la verifica di attestazioni verificabili che fungono da prova di conformità a profili o standard specifici. Ciò consente alle entità all'interno della federazione di dimostrare l'aderenza agli standard di sicurezza, privacy e operativi concordati.
FR13	**Meccanismo Decentralizzato per la Risoluzione delle Controversie**: progettare un meccanismo decentralizzato per la risoluzione delle controversie che consenta ai membri della federazione di verificare in modo indipendente la creazione di trust storica e gli artefatti firmati, riducendo la dipendenza dalle autorità centrali e semplificando il processo di risoluzione.
FR14	**Interoperabilità tra Federazioni**: garantire che il sistema sia in grado di interoperare con altre federazioni o Trust Framework, facilitando transazioni e creazione di trust tra federazioni senza compromettere la sicurezza o la conformità.
FR15	**Organismi di Registrazione Autonomi**: il sistema deve facilitare l'integrazione di organismi di registrazione autonomi che operano in conformità con le regole della federazione. Questi organismi hanno il compito di valutare e registrare le entità all'interno della federazione, secondo le regole prestabilite e la loro conformità che deve essere periodicamente attestata.
FR16	**Audit Periodico degli Organismi di Registrazione e delle Entità**: implementare meccanismi per l'audit periodico e il monitoraggio dello stato di conformità sia degli organismi di registrazione che delle loro entità registrate.
FR17	**Attestazione di Conformità per i Dispositivi Personali**: organismi fidati, in forma di entità di federazione, dovrebbero emettere attestazioni di conformità e fornire prove firmate di tale conformità per l'hardware dei dispositivi personali utilizzati all'interno della federazione. Queste attestazioni dovrebbero essere attestate e periodicamente rinnovate per garantire che i dispositivi soddisfino gli attuali standard di sicurezza.
FR18	**Monitoraggio Automatizzato della Conformità**: il sistema dovrebbe includere strumenti automatizzati per il monitoraggio della conformità delle entità agli standard della federazione. Questa automazione aiuta nel rilevamento precoce di potenziali problemi di conformità.

continues on next page

Tabella 6.3 – continua dalla pagina precedente

ID	Descrizione
FR19	**Binding Sicuro delle Capacità del Protocollo**: il protocollo sicuro deve consentire lo scambio di dati sulle capacità specifiche del protocollo come metadati crittograficamente vincolati allegati a un'identità specifica. Questi metadati dovrebbero definire le capacità tecniche associate all'identità, garantendo prova verificabile e associazione antimanomissione per una robusta creazione di trust e controllo degli accessi.

6.5 Endpoint API di Federazione

OpenID Federation 1.0 utilizza Servizi Web RESTful protetti su HTTPs. OpenID Federation 1.0 definisce quali sono gli endpoint web che i partecipanti DEVONO rendere pubblicamente disponibili. La tabella sottostante riassume gli endpoint e i loro ambiti.

Tutti gli endpoint elencati di seguito sono definiti nelle specifiche OID-FED.

nome endpoint	richiesta http	ambito	richiesto per
federation metadata	**GET** .well-known/openid-federation	Metadati che un'Entità pubblica su se stessa, verificabili con una terza parte fidata (Entità Superiore). È chiamata Entity Configuration.	Trust Anchor, Intermediario, Fornitore di Wallet, Relying Party, Credential Issuer
subordinate list endpoint	**GET** /list	Elenca i Subordinati. Vedi OID-FED Sezione 8.2	Trust Anchor, Intermediario
fetch endpoint	**GET** /federation_fetch_endpoint?sub=https://rp.example.org	Restituisce un JWT firmato su un soggetto specifico, il suo Subordinato. È chiamato Subordinate Statement. Vedi OID-FED Sezione 8.1	Trust Anchor, Intermediario
trust mark status	**POST** /trust_mark_status	Restituisce lo stato dell'emissione (validità) di un Trust Mark relativo a un soggetto specifico. Vedi OID-FED Sezione 8.4	Trust Anchor, Intermediario

continues on next page

Tabella 6.4 – continua dalla pagina precedente

nome endpoint	richiesta http	ambito	richiesto per
trust mark list	**GET** /trust_mark_listing?trust_mark_type=...	Elenca tutte le entità per le quali sono stati emessi Trust Mark e sono ancora validi. Vedi OID-FED Sezione 8.5	Trust Anchor, Intermediario
trust mark	**GET** /trust_mark?trust_mark_type=...	Restituisce il Trust Mark relativo a un soggetto specifico. Vedi OID-FED Sezione 8.6	Trust Anchor, Intermediate
historical keys	**GET** /federation_historical_keys	Elenca le chiavi scadute e revocate, con la motivazione della revoca. Vedi OID-FED Sezione 8.7	Trust Anchor, Intermediario
subordinate events	**GET** /federation_subordinate_events_endpoint?sub=https://rp.example.org	Restituisce una traccia storica degli eventi di registrazione sui Subordinati Immediati, come registrazione, revoca e aggiornamenti delle loro Chiavi dell'Entità di Federazione. Vedi la sezione *Federation Subordinate Events Endpoint* (pagina 156) per maggiori dettagli.	Trust Anchor, Intermediario

Tutte le risposte degli endpoint di federazione sono sotto forma di JWT firmato, ad eccezione dell'endpoint di Elenco Subordinati e dell'endpoint di Stato Trust Mark che sono serviti come JSON semplice per impostazione predefinita. L'Endpoint Eventi Subordinati della Federazione restituisce anche JWT firmati con il tipo di contenuto `application/entity-events-statement+jwt`.

6.6 Configurazione della Federazione

La configurazione della federazione è pubblicata dal Trust Anchor all'interno della sua Entity Configuration, è disponibile nel path web well-known corrispondente a **.well-known/openid-federation**.

Tutti i partecipanti nella federazione DEVONO ottenere la configurazione di federazione prima di entrare nella fase operativa, e DEVONO mantenerla aggiornata. La configurazione di federazione è l'Entity Configuration del Trust Anchor, contiene le chiavi pubbliche per le operazioni di firma.

Di seguito è riportato un esempio non normativo di un'Entity Configuration del Trust Anchor, dove ogni parametro è documentato nella specifica OpenID Federation:

```
{
    "alg": "ES256",
    "kid": "FifYx03bnosD8m6gYQIfNHNP9cM_Sam9Tc5nLloIIrc",
    "typ": "entity-statement+jwt"
}
```

```
  {
    "exp": 1649375259,
    "iat": 1649373279,
    "iss": "https://trust-anchor.eid-wallet.example.it",
    "sub": "https://trust-anchor.eid-wallet.example.it",
    "jwks": {
        "keys": [
            {

                "kty": "EC",
                "kid": "X2ZOMHNGSDc4ZlBrcXhMT3MzRmRZOG9Jd3o2QjZDam51cUhhUFRuOWd0WQ",
                "crv": "P-256",
                "x": "1kNR9Ar3MzMokYTY8BRvRIue85NIXrYX4XD3K4JW7vI",
                "y": "slT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrY",
                "x5c": [
                  // <self-issued X.509 certificate about the Trust Anchor>
                  ]
            }
        ]
    },
    "metadata": {
        "federation_entity": {
            "organization_name": "example TA",
            "contacts":[
                "tech@eid.trust-anchor.example.eu"
            ],
            "homepage_uri": "https://trust-anchor.eid-wallet.example.it",
            "logo_uri":"https://trust-anchor.eid-wallet.example.it/static/svg/logo.svg",
```

```
            "federation_fetch_endpoint": "https://trust-anchor.eid-wallet.example.it/fetch",
            "federation_resolve_endpoint": "https://trust-anchor.eid-wallet.example.it/resolve"
,
            "federation_list_endpoint": "https://trust-anchor.eid-wallet.example.it/list",
            "federation_trust_mark_status_endpoint": "https://trust-anchor.eid-wallet.example.i
t/trust_mark_status",
            "federation_trust_mark_list_endpoint": "https://trust-anchor.eid-wallet.example.it/
trust_mark_listing",
            "federation_subordinate_events_endpoint": "https://trust-anchor.eid-wallet.example.
it/events"
        }
    },
    "trust_mark_issuers": {
        "https://trust-anchor.eid-wallet.example.it/openid_relying_party/public": [
            "https://trust-anchor.eid-wallet.example.it",
            "https://public.intermediary.other.org"
        ],
        "https://trust-anchor.eid-wallet.example.it/openid_relying_party/private": [
            "https://private.other.intermediary.org"
        ]
    }
}
```


6.7 Entity Configuration

L'Entity Configuration è il documento verificabile che ogni Entità di Federazione DEVE pubblicare per proprio conto, nell'endpoint **.well-known/openid-federation**.

La Risposta HTTP dell'Entity Configuration DEVE impostare il tipo di media su *application/entity-statement+jwt*.

L'Entity Configuration DEVE essere firmata crittograficamente. La parte pubblica di questa chiave DEVE essere fornita nell'Entity Configuration e all'interno del Subordinate Statement emesso da un superiore immediato e relativo alla sua Entità di Federazione subordinata.

L'Entity Configuration PUÒ anche contenere uno o più Trust Mark.

Ruolo nell'Onboarding: Le nuove entità pubblicano la loro Entity Configuration come parte del loro processo di registrazione, dichiarando le loro capacità, protocolli supportati e stato di conformità alla federazione. La configurazione serve come dichiarazione iniziale dell'entità della sua prontezza tecnica e ambito operativo, consentendo ad altri partecipanti di scoprire e validare il suo stato di registrazione.

Ruolo nelle Operazioni: Durante le operazioni di credenziali, le Entity Configuration sono recuperate da wallet, credential issuer e relying party per verificare lo stato operativo corrente, le capacità supportate e le attestazioni di conformità di altre entità. Questo abilita la scoperta

dinamica degli endpoint di servizio, delle chiavi crittografiche e delle versioni di protocollo richieste per lo scambio sicuro di credenziali.

I dettagli tecnici sull'Entity Configuration del Fornitore di Wallet, Credential Issuer e Relying Party sono forniti rispettivamente nelle Sezioni *Entity Configuration del Fornitore di Wallet* (pagina 326), credential-issuer-entity-configuration:Entity Configuration del Fornitore di Attestati Elettronici e *Entity Configuration di una Relying Party* (pagina 373).

Nota

Firma dell'Entity Configuration

Tutte le operazioni di controllo della firma riguardanti le Entity Configuration, i Subordinate Statement e i Trust Mark, sono eseguite con le chiavi pubbliche di Federazione. Per gli algoritmi supportati fare riferimento alla Sezione *Algoritmo Crittografico*.

6.7.1 Parametri Comuni delle Entity Configuration

Le Entity Configuration di tutti i partecipanti nella federazione DEVONO avere in comune i parametri elencati di seguito.

Claim	Descrizione
iss	String. Identificatore dell'Entità emittente.
sub	String. Identificatore dell'Entità a cui si riferisce. DEVE essere uguale a `iss`.
iat	UNIX Timestamp con il tempo di generazione del JWT, codificato come NumericDate come indicato in **RFC 7519**.
exp	UNIX Timestamp con il tempo di scadenza del JWT, codificato come Numeric Date come indicato in **RFC 7519**.
jwks	Un JSON Web Key Set (JWKS) **RFC 7517** che rappresenta la parte pubblica delle chiavi di firma dell'Entità in questione. Ogni JWK nel set JWK DEVE avere un ID chiave (claim kid) e PUÒ avere un parametro *x5c*, come definito in **RFC 7517**. Contiene le Chiavi dell'Entità di Federazione richieste per le operazioni di Trust Evaluation. **x5c**: Il parametro *x5c* incluso nel parametro *jwks* dell'Entity Configuration DEVE contenere solo il Certificato X.509 auto-emesso relativo al corrispondente *jwk*.
metadata	Oggetto JSON. Ogni chiave dell'Oggetto JSON rappresenta un identificatore di tipo di metadati contenente un Oggetto JSON che rappresenta i metadati, secondo lo schema di metadati di quel tipo. Un'Entity Configuration PUÒ contenere più dichiarazioni di metadati, ma solo una per ogni tipo di metadati (<**entity_type**>). i tipi di metadati sono definiti nella sezione Tipi di Metadati.

6.7.2 Entity Configuration Trust Anchor

L'Entity Configuration del Trust Anchor, oltre ai parametri comuni elencati sopra, utilizza i seguenti parametri:

Claim	Descrizione	Richiesto
trust_mark_issuers	Array JSON che definisce quali autorità di Federazione sono considerate affidabili per l'emissione di Trust Mark specifici, assegnati con i loro identificatori unici.	✓

6.7.3 Trust Mark degli Intermediari di Relying Party

Nell'ecosistema IT-Wallet, gli Intermediari di Relying Party che agiscono per conto dei Relying Party (ai sensi dell'Art. 5b(8) del Regolamento eIDAS2 EU_2024_1183) sono soggetti a uno specifico regime di Trust Mark che consente al Wallet di riconoscere e mostrare all'Utente che il RP richiedente opera tramite un Intermediario di Relying Party.

Emissione del Trust Mark all'Intermediario di Relying Party

Il Trust Anchor DEVE includere gli Intermediari di Relying Party nell'attributo `trust_mark_issuers` della propria Entity Configuration, autorizzandoli ad emettere Trust Mark per le Foglie a loro afferenti. Il Trust Anchor emette all'Intermediario di Relying Party un Trust Mark che attesta il suo ruolo, utilizzando `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier_intermediary` come `trust_mark_type` per distinguerlo dalle altre entità presenti nell'ecosistema.

L'Intermediario di Relying Party DEVE includere questo Trust Mark nella propria Entity Configuration come prova del suo ruolo riconosciuto all'interno della Federazione.

Emissione di Trust Mark alle Foglie afferenti all'Intermediario di Relying Party

I Trust Mark emessi alle Foglie (Relying Party) afferenti a un Intermediario di Relying Party DEVONO essere emessi dall'Intermediario di Relying Party stesso, e non direttamente dal Trust Anchor.

Un Relying Party afferente a un Intermediario di Relying Party DEVE includere nella propria Entity Configuration:

- Il Trust Mark emesso dall'Intermediario di Relying Party, come prova del completamento del processo di onboarding gestito dall'Intermediario di Relying Party;

- Il proprio `authority_hints` contenente l'URL dell'Intermediario di Relying Party (non direttamente del Trust Anchor), indicando così la corretta catena gerarchica.

Di seguito è riportato un esempio non normativo di Entity Configuration del Trust Anchor che mostra la configurazione aggiornata di `trust_mark_issuers` con gli Intermediari di Relying Party autorizzati:

```
{
  "trust_mark_issuers": {
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_verifier": [
      "https://trust-anchor.eid-wallet.example.it",
      "https://intermediary.example.org"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_verifier_intermediary": [
      "https://trust-anchor.eid-wallet.example.it"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_credential
_issuer": [
      "https://trust-anchor.eid-wallet.example.it"
    ],
    "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/openid_wallet_pro
vider": [
      "https://trust-anchor.eid-wallet.example.it"
    ]
  }
}
```

Nota

Il Trust Mark con identificativo che termina con `federation-entity/openid_credential_verifier_intermediary` è emesso esclusivamente dal Trust Anchor ed è quello che il Wallet utilizza per verificare che un'entità sia un Intermediario di Relying Party riconosciuto. I Trust Mark `https://<federation_authority_domain>/trust_marks/federation-entity/openid_credential_verifier` DOVREBBERO invece essere emessi anche dagli Intermediari di Relying Party per le Foglie a loro afferenti.

6.7.4 Entity Configuration Foglie e Intermediari

Oltre ai claim precedentemente definiti, l'Entity Configuration delle Foglie e delle Entità Intermedie utilizza i seguenti parametri:

Claim	Descrizione	Richiesto
authority_hints	Array di URL (String). Contiene un elenco di URL delle entità superiori immediate, come il Trust Anchor o un Intermediario, che emette un Subordinate Statement relativo a questo soggetto.	✓
trust_marks	Un Array JSON contenente i Trust Mark.	✓

6.7.5 Tipi di Metadati

In questa sezione sono definiti i principali tipi di metadati mappati sui ruoli dell'ecosistema, fornendo i riferimenti del protocollo di metadati per ognuno di questi.

Nota

Le voci che non hanno alcun riferimento a una bozza o standard noto sono intese per essere definite in questo riferimento tecnico.

Entità OpenID	Entità EUDI	Tipo di Metadati	Riferimenti
Trust Anchor	Trust Anchor	`federation_entity`	OID-FED
Intermediario	Intermediario	`federation_entity`	OID-FED
Fornitore di Wallet	Fornitore di Wallet	`federation_entity`, `openid_wallet_provider`	OID-FED-WALLET
Authorization Server		`federation_entity`, `oauth_authorization_server`	OPENID4VCI

continues on next page

Tabella 6.8 – continua dalla pagina precedente

Entità OpenID	Entità EUDI	Tipo di Metadati	Riferimenti
Credential Issuer	Fornitore PID, Fornitore (Q)EAA	`federation_entity`, `openid_credential_issuer`, [`oauth_authorization_server`]	OPENID4VCI
Relying Party	Relying Party	`federation_entity`, `openid_credential_verifier`	OID-FED, OpenID4VP

Nota

I metadati del Fornitore di Wallet sono definiti nella sezione sottostante.

Soluzione Wallet (pagina 287).

Nota

Nei casi in cui un Fornitore PID/EAA implementa sia il Credential Issuer che l'Authorization Server, DEVE incorporare sia `oauth_authorization_server` che `openid_credential_issuer` all'interno dei suoi tipi di metadati. Altre implementazioni possono dividere il Credential Issuer dall'Authorization Server, quando questo accade i metadati del Credential Issuer DEVONO contenere i parametri *authorization_servers*, incluso l'identificatore unico dell'Authorization Server. Inoltre, se dovesse esserci la necessità di Autenticazione dell'Utente da parte del Credential Issuer, potrebbe essere necessario includere il tipo di metadati pertinente, sia `openid_relying_party` che `openid_credential_verifier`.

6.8 Metadati di federation_entity Foglie

I metadati *federation_entity* per le Foglie contiene i seguenti claim.

Claim	Descrizione
organization_name	OBBLIGATORIO. Vedi OID-FED Sezione 5.2.2
homepage_uri	OBBLIGATORIO. OBBLIGATORIO se `organization_uri` è assente.

continues on next page

Tabella 6.9 – continua dalla pagina precedente

Claim	Descrizione
organization_uri	CONDIZIONALE. OBBLIGATORIO se `homepage_uri` è assente. Vedi OID-FED Sezione 5.2.2
policy_uri	OBBLIGATORIO. Vedi OID-FED Sezione 5.2.2
logo_uri	OBBLIGATORIO. URL del logo dell'entità; DEVE essere in formato SVG. Vedi OID-FED Sezione 5.2.2
contacts	OBBLIGATORIO. Indirizzo email verificato istituzionale (PEC) dell'entità. Vedi OID-FED Sezione 5.2.2
federation_resolve_endpoint	OPZIONALE. Vedi OID-FED Sezione 8.3
tos_uri	OPZIONALE. Stringa URL che punta a un documento di termini di servizio leggibile dall'uomo per il client che descrive una relazione contrattuale tra l'utente finale e il client che l'utente finale accetta quando autorizza il client. Vedi OID-FED.

6.9 I Subordinate Statement

I Trust Anchor e gli Intermediari pubblicano Subordinate Statement relativi ai loro Subordinati immediati. Il Subordinate Statement PUÒ contenere una policy di metadati e i Trust Mark relativi a un Subordinato.

La policy di metadati, quando applicata, apporta una o più modifiche ai metadati finali della Foglia. I metadati finali di una Foglia sono derivati dalla Trust Chain che contiene tutte le dichiarazioni, a partire dall'Entity Configuration fino al Subordinate Statement emesso dal Trust Anchor.

I Trust Anchor e gli Intermediari DEVONO esporre l'endpoint Federation Fetch, dove i Subordinate Statement sono richiesti per validare la firma dell'Entity Configuration della Foglia.

Nota

L'endpoint Federation Fetch PUÒ anche pubblicare certificati X.509 per ognuna delle chiavi pubbliche del Subordinato. Rendendo la distribuzione dei certificati X.509 emessi tramite un servizio RESTful.

Ruolo nell'Onboarding: Durante la registrazione delle entità, i Trust Anchor e gli Intermediari emettono Subordinate Statement per attestare formalmente la registrazione e le capacità delle nuove entità. Queste dichiarazioni stabiliscono la relazione di trust gerarchica e applicano eventuali policy di metadati richieste che vincolano o migliorano le capacità dichiarate dell'entità basate sulle policy di federazione.

Ruolo nelle Operazioni: Durante le operazioni di credenziali, i Subordinate Statement sono recuperati per validare le catene di trust e applicare le policy di metadati correnti. Abilitano la verifica in tempo reale dello stato di registrazione di un'entità e garantiscono che le capacità operative siano conformi alle policy a livello di federazione e all'ambito autorizzato dell'entità.

Di seguito è riportato un esempio non normativo di un Subordinate Statement emesso da un Registration Body (come il Trust Anchor o il suo Intermediario) in relazione a uno dei suoi Subordinati.

```
{
    "alg": "ES256",
    "kid": "em3cmnZgHIYFsQ090N6B3Op7LAAqj8rghMhxGmJstqg",
    "typ": "entity-statement+jwt"
}

{
    "exp": 1649623546,
    "iat": 1649450746,
    "iss": "https://intermediate.example.org",
    "sub": "https://rp.example.it",
    "jwks": {
        "keys": [ // keys about the Subordinate
            {
                "kty": "EC",
                "kid": "2HnoFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "crv": "P-256",
                "x": "1kNR9Ar3MzMokYTY8BRvRIue85NIXrYX4XD3K4JW7vI",
                "y": "slT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrY",
                "x5c": [
                  // <X.509 certificate about the Subordinate>
                  ]
            }
        ]
    },
    "metadata_policy": {
        "openid_credential_verifier": {
            "scope": {
                "subset_of": [
                     "eu.europa.ec.eudiw.pid.1",
                     "given_name",
                     "family_name",
                     "email"
                  ]
            },
            "vp_formats": {
                "dc+sd-jwt": {
                    "sd-jwt_alg_values": [
```

```
                    "ES256",
                    "ES384"
                ],
                "kb-jwt_alg_values": [
                    "ES256",
                    "ES384"
                ]
            }
        }
    }
}
```

Nota

Firma del Subordinate Statement

Le stesse considerazioni e requisiti fatti per l'Entity Configuration e in relazione ai meccanismi di firma DEVONO essere applicati per i Subordinate Statement.

6.9.1 Subordinate Statement

Il Subordinate Statement emesso dai Trust Anchor e dagli Intermediari contiene i seguenti attributi:

Claim	Descrizione	Richiesto
iss	Vedi OID-FED Sezione 3 per ulteriori dettagli.	✓
sub	Vedi OID-FED Sezione 3 per ulteriori dettagli.	✓
iat	Vedi OID-FED Sezione 3 per ulteriori dettagli.	✓
exp	Vedi OID-FED Sezione 3 per ulteriori dettagli.	✓
jwks	JWKS di Federazione dell'entità *sub*. Vedi OID-FED Sezione 3 per ulteriori dettagli.	✓
metadata_ policy	Oggetto JSON che descrive la policy dei Metadati. Ogni chiave dell'Oggetto JSON rappresenta un identificatore del tipo di metadati e ogni valore DEVE essere un Oggetto JSON che rappresenta la policy dei metadati secondo quel tipo di metadati. Si prega di fare riferimento alle specifiche OID-FED, Sezione 6.1, per i dettagli di implementazione.	✗

continues on next page

Tabella 6.10 – continua dalla pagina precedente

Claim	Descrizione	Richiesto
trust_marks	Array JSON contenente i Trust Mark emessi da se stesso per il soggetto subordinato.	✗
constraints	PUÒ contenere gli **allowed_leaf_entity_types**, che restringe quali tipi di metadati il soggetto è autorizzato a pubblicare. PUÒ contenere il numero massimo di Intermediari consentiti tra se stesso e la Foglia (**max_path_length**)	✓

6.10 Discovery della Federazione

La Scoperta di Entità di Federazione è il processo attraverso il quale i partecipanti verificano l'identità e lo stato di altre entità di federazione. Questo comporta l'interrogazione degli endpoint di federazione per confermare lo stato di validità dell'entità e la conformità al Trust Framework.

Il processo di scoperta utilizza l'API di Federazione per raccogliere informazioni sulle entità e produce Trust Chain che validano la conformità e l'autorizzazione dell'entità all'interno della federazione.

Il processo di scoperta stabilisce i concetti fondamentali che vengono poi applicati in scenari operativi specifici come dettagliato nella sezione Meccanismo di Trust Evaluation.

Nota

I Trust Anchor DEVONO distribuire le loro Chiavi Pubbliche di Federazione attraverso meccanismi sicuri fuori banda, come pubblicarle su una pagina web verificata o archiviarle in un repository remoto come parte di una lista di trust. La logica dietro questo requisito è che fare affidamento solo sui dati forniti all'interno dell'Entity Configuration del Trust Anchor non mitiga adeguatamente i rischi associati agli attacchi di manipolazione DNS e TLS. Per garantire la sicurezza, tutti i partecipanti DEVONO ottenere le chiavi pubbliche del Trust Anchor utilizzando questi metodi fuori banda. Dovrebbero quindi confrontare queste chiavi con quelle ottenute dall'Entity Configuration del Trust Anchor, scartando qualsiasi chiave che non corrisponda. Questo processo aiuta a garantire l'integrità e l'autenticità delle chiavi pubbliche del Trust Anchor e la sicurezza generale della federazione (*WP_017* (pagina 781)).

Ogni Subordinate Statement è verificabile nel tempo e DEVE avere una data di scadenza. La revoca di ogni dichiarazione è verificabile in tempo reale e online (solo per flussi remoti) attraverso gli endpoint di federazione.

Nota

La revoca di un'Entità è fatta con l'indisponibilità del Subordinate Statement relativo ad essa. Se il Trust Anchor o il suo Intermediario non pubblica un Subordinate Statement valido, o se pubblica un Subordinate Statement scaduto/non valido, il soggetto del Subordinate Statement DEVE essere inteso come non valido o revocato.

La concatenazione delle dichiarazioni, attraverso la combinazione di questi meccanismi di firma e il binding di claim e chiavi pubbliche, forma la Trust Chain.

Le Trust Chain possono anche essere verificate offline, utilizzando una delle chiavi pubbliche del Trust Anchor.

Nota

Poiché l'Istanza del Wallet non è un'Entità di Federazione, il Meccanismo di Trust Evaluation relativo ad essa **richiede la presentazione del Wallet Attestation durante le fasi di emissione e presentazione delle credenziali**.

Il Wallet Attestation trasmette tutte le informazioni richieste riguardanti l'istanza, come la sua chiave pubblica e qualsiasi altra informazione tecnica o amministrativa, senza alcun dato personale dell'Utente.

6.11 Trust Chain

La Trust Chain è una sequenza di dichiarazioni verificate che valida la conformità di un partecipante con la Federazione. Ha una data di scadenza, oltre la quale DEVE essere rinnovata per ottenere i metadati freschi e aggiornati. La data di scadenza della Trust Chain è determinata dal timestamp di scadenza più precoce tra tutti i timestamp di scadenza contenuti nelle dichiarazioni. Nessuna Entità può forzare la data di scadenza della Trust Chain ad essere superiore a quella configurata dal Trust Anchor.

Ruolo nell'Onboarding: Durante la registrazione delle entità, le Trust Chain sono costruite per dimostrare la relazione di trust gerarchica completa dal Trust Anchor alla nuova entità. Questo stabilisce la posizione legittima dell'entità nella federazione e valida la sua conformità con tutte le policy e i vincoli applicabili.

Ruolo nelle Operazioni: Durante l'emissione e la presentazione delle credenziali, le Trust Chain forniscono prova crittografica della validità dell'entità e dello stato di conformità. Abilitano la verifica offline delle relazioni di trust e supportano scenari dove l'accesso in tempo reale agli endpoint di federazione potrebbe non essere disponibile, garantendo al contempo che le attestazioni di trust rimangano correnti e verificabili.

Di seguito è riportata una rappresentazione astratta di una Trust Chain.

```
[
    "EntityConfiguration-as-SignedJWT-selfissued-byLeaf",
    "EntityStatement-as-SignedJWT-issued-byTrustAnchor"
]
```

Di seguito è riportato un esempio non normativo di una Trust Chain, composta da un Array JSON contenente JWT, con un Intermediario coinvolto.

```
[
  "eyJhbGciOiJFUzI1NiIsImtpZCI6Ik5GTTFXVVZpVWxZelVXcExhbWxmY0VwUFJWWTJWWFpJUmpCblFYWm1SSGhLWVVW
WVVsZFRRbkEyTkEiLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p3dCJ9.eyJleHAiOjE2NDk1OTA2MD
IsImlhdCI6MTY0OTQxNzg2MiwiaXNzIjoiaHR0cHM6Ly9ycC5leGFtcGxlLm9yZyIsInN1YiI6Imh0dHBzOi8vcnAuZXhhb
XBsZS5vcmciLCJqd2tzIjp7ImtleXMiOlt7Imt0eSI6IkVDIiwia2lkIjoiTkZNMVdVVmlVbFl6VVdwTGFtbGZjRXBQUlZZ
MlZYWklSakJuUVhabVJIaEtZVVZZVWxkVFFuQTJOQSIsImNydiI6IlAtMjU2IiwieCI6InVzbEMzd2QtcFgzd3o0YlJZbnd
5M2x6cGJHWkZoTjk2aEwyQUhBM01RNlkiLCJ5IjoiVkxDQlhGV2xkTlNOSXo4a0gyOXZMUjROMThCa3dHT1gyNnpRb3J1UT
FNNCJ9XX0sIm1ldGFkYXRhIjp7Im9wZW5pZF9yZWx5aW5nX3BhcnR5Ijp7ImFwcGxpY2F0aW9uX3R5cGUiOiJ3ZWIiLCJjb
GllbnRfaWQiOiJodHRwczovL3JwLmV4YW1wbGUub3JnLyIsImNsaWVudF9yZWdpc3RyYXRpb25fdHlwZXMiOlsiYXV0b21h
dGljIl0sImp3a3MiOnsia2V5cyI6W3sia3R5IjoiRUMiLCJraWQiOiJORk0xV1VWaVVsWXpVV3BMYW1sZmNFcFBSVlkyVlh
aSVJqQm5RWFptUkhoS1lVVllVbGRUUW5BMk5BIiwiY3J2IjoiUC0yNTYiLCJ4IjoidXNsQzN3ZC1wWDN3ejRiUllud3kzbH
pwYkdaRmhOOTZoTDJBSEEzTVE2WSIsInkiOiJWTENCWEZXbGROU05JejhrSDI5dkxSNE4xOEJrd0dPWDI2elFvcnVRMU00I
n1dfSwiY2xpZW50X25hbWUiOiJOYW1lIG9mIGFuIGV4YW1wbGUgb3JnYW5pemF0aW9uIiwiY29udGFjdHMiOlsib3BzQHJw
LmV4YW1wbGUuaXQiXSwiZ3JhbnRfdHlwZXMiOlsicmVmcmVzaF90b2tlbiIsImF1dGhvcml6YXRpb25fY29kZSJdLCJyZWR
pcmVjdF91cmlzIjpbImh0dHBzOi8vcnAuZXhhbXBsZS5vcmcvb2lkYy9ycC9jYWxsYmFjay8iXSwicmVzcG9uc2VfdHlwZX
MiOlsiY29kZSJdLCJzY29wZSI6ImV1LmV1cm9wYS5lYy5ldWRpdy5waWQuMSBldS5ldXJvcGEuZWMuZXVkaXcucGlkLml0L
jEgZW1haWwiLCJzdWJqZWN0X3R5cGUiOiJwYWlyd2lzZSJ9LCJmZWRlcmF0aW9uX2VudGl0eSI6eyJmZWRlcmF0aW9uX3Jl
c29sdmVfZW5kcG9pbnQiOiJodHRwczovL3JwLmV4YW1wbGUub3JnL3Jlc29sdmUvIiwib3JnYW5pemF0aW9uX25hbWUiOiJ
FeGFtcGxlIFJQIiwiaG9tZXBhZ2VfdXJpIjoiaHR0cHM6Ly9ycC5leGFtcGxlLm0IiwicG9saWN5X3VyaSI6Imh0dHBzOi
8vcnAuZXhhbXBsZS5pdC9wb2xpY3kiLCJsb2dvX3VyaSI6Imh0dHBzOi8vcnAuZXhhbXBsZS5pdC9zdGF0aWMvbG9nby5zd
mciLCJjb250YWN0cyI6WyJ0ZWNoQGV4YW1wbGUuaXQiXX19LCJ0cnVzdF9tYXJrcyI6W3siaWQiOiJodHRwczovL3JlZ2lz
dHJ5LmVpZGFzLnRydXN0LWFuY2hvci5leGFtcGxlLmV1L29wZW5pZF9yZWx5aW5nX3BhcnR5L3B1YmxpYy8iLCJ0cnVzdF9
tYXJrIjoiZXlKaCBcdTIwMjYifV0sImF1dGhvcml0eV9oaW50cyI6WyJodHRwczovL2ludGVybWVkaWF0ZS5laWRhcy5leG
FtcGxlLm9yZyJdfQ.Un315HdckvhYA-iRregZAmL7pnfjQH2APz82blQO5S0sl1JR0TEFp5E1T913g8GnuwgGtMQUqHPZwV
6BvTLA8g",
```

”eyJhbGciOiJFUzI1NiIsImtpZCI6IlNURkRXV2hKY0dWWFgzQjNSVmRaYWtsQ0xUTnVNa000WTNGNlFUTk9kRXRyZFhG
WVlYWjJjWGN0UVEiLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p3dCJ9.eyJleHAiOjE2NDk2MjM1ND
YsImlhdCI6MTY0OTQ1MDc0NiwiaXNzIjoiaHR0cHM6Ly9pbnRlcm1lZGlhdGUuZWlkYXMuZXhhbXBsZS5vcmciLCJzdWIiO
iJodHRwczovL3JwLmV4YW1wbGUub3JnIiwiandrcyI6eyJrZXlzIjpbeyJrdHkiOiJFQyIsImtpZCI6Ik5GTTFXVVZpVWxZ
elVXcExhbWxmY0VwUFJWWTJWWFpJUmpCblFYWm1SSGhLWVVWWVVsZFRRbkEyTkEiLCJjcnYiOiJQLTI1NiIsIngiOiJ1c2x
DM3dkLXBYM3d6NGJSWW53eTNsenBiR1pGaE45NmhMMkFIQTNNUTZZIiwieSI6IlZMQ0JYRldsZE5TTkl6OGtIMjl2TFI0Tj
E4Qmt3R09YMjZ6UW9ydVExTTQifV19LCJtZXRhZGF0YV9wb2xpY3kiOnsib3BlbmlkX3JlbHlpbmdfcGFydHkiOnsic2Nvc
GUiOnsic3Vic2V0X29mIjpbImV1LmV1cm9wYS5lYy5ldWRpdy5waWQuMSwgIGV1LmV1cm9wYS5lYy5ldWRpdy5waWQuaXQu
MSJdfSwicmVxdWVzdF9hdXRoZW50aWNhdGlvbl9tZXRob2RzX3N1cHBvcnRlZCI6eyJvbmVfb2YiOlsicmVxdWVzdF9vYmp
lY3QiXX0sInJlcXVlc3RfYXV0aGVudGljYXRpb25fc2lnbmluZ19hbGdfdmFsdWVzX3N1cHBvcnRlZCI6eyJzdWJzZXRfb2
YiOlsiUlMyNTYiLCJSUzUxMiIsIkVTMjU2IiwiRVM1MTIiLCJQUzI1NiIsIlBTNTEyIl19fX0sInRydXN0X21hcmtzIjpbe
yJpZCI6Imh0dHBzOi8vdHJ1c3QtYW5jaG9yLmV4YW1wbGUuZXUvb3BlbmlkX3JlbHlpbmdfcGFydHkvcHVibGljLyIsInRy
dXN0X21hcmsiOiJleUpoYiBcdTIwMjYifV19._qt5-T6DahP3TuWa_27klE8I9Z_sPK2FtQlKY6pGMPchbSI2aHXY3aAXDU
rObPo4CHtqgg3J2XcrghDFUCFGEQ”,”eyJhbGciOiJFUzI1NiIsImtpZCI6ImVXa3pUbWt0WW5kblZHMWxhMjU1ZDJkQ2RV
ZERSazQwUWt0WVlVMWFhRFZYT1RobFpHdFdXSGQ1WnciLCJ0eXAiOiJhcHBsaWNhdGlvbi9lbnRpdHktc3RhdGVtZW50K2p
3dCJ9.eyJleHAiOjE2NDk2MjM1NDYsImlhdCI6MTY0OTQ1MDc0NiwiaXNzIjoiaHR0cHM6Ly90cnVzdC1hbmNob3IuZXhhb
XBsZS5ldSIsInN1YiI6Imh0dHBzOi8vaW50ZXJtZWRpYXRlLmVpZGFzLmV4YW1wbGUub3JnIiwiandrcyI6eyJrZXlzIjpb
eyJrdHkiOiJFQyIsImtpZCI6IlNURkRXV2hKY0dWWFgzQjNSVmRaYWtsQ0xUTnVNa000WTNGNlFUTk9kRXRyZFhGWVlYWjJ
jWGN0UVEiLCJjcnYiOiJQLTI1NiIsIngiOiJyQl9BOGdCUnh5NjhVTkxZRkZLR0ZMR2VmWU5XYmgtSzh1OS1GYlQyZkZJIi
wieSI6IlNuWVk2Y3NjZnkxcjBISFhLTGJuVFZsamFndzhOZzNRUEs2WFVoc2UzdkUifV19LCJ0cnVzdF9tYXJrcyI6W3sia
WQiOiJodHRwczovL3RydXN0LWFuY2hvci5leGFtcGxlLmV1L2ZlZGVyYXRpb25fZW50aXR5L3RoYXQtcHJvZmlsZSIsInRy
dXN0X21hcmsiOiJleUpoYiBcdTIwMjYifV19.r3uoi-U0tx0gDFlnDdITbcwZNUpy7M2tnh08jlD-Ej9vMzWMCXOCCuwIn0
ZT0jS4M_sHneiG6tLxRqj-htI70g”
]

Nota

L’intera Trust Chain è verificabile possedendo solo le chiavi pubbliche del Trust Anchor.

Ci sono eventi in cui le chiavi non sono disponibili per verificare l’intera catena di trust:

- **Cambio di Chiave da parte del Credential Issuer**: Il Credential Issuer PUÒ aggiornare le sue chiavi crittografiche. Le chiavi crittografiche DEVONO essere considerate valide se valutate entro il loro periodo di validità originariamente designato a meno che una ragione di sicurezza non le renda inutilizzabili. La ragione di revoca DEVE essere pubblicata. Le chiavi crittografiche storiche, cioè le chiavi crittografiche pubbliche inutilizzate o revocate, DEVONO essere pubblicate utilizzando l’Endpoint delle Chiavi Storiche di Federazione.
- **Cambio nei Tipi di Credenziali**: Se il Credential Issuer cambia i **tipi** di Credenziali emesse, per esempio decidendo di non emettere più uno o più tipi di Credenziali, le relative chiavi crittografiche pubbliche DEVONO essere disponibili per il periodo di validità originariamente designato.
- **Fusione di Credential Issuer**: Se un Credential Issuer si fonde con un altro, creando una

nuova Entità Organizzativa o lavorando per conto di un'altra utilizzando un hostname o dominio diverso, la configurazione di federazione **precedentemente** disponibile e le chiavi storiche DEVONO essere mantenute disponibili agli endpoint well-known del Credential Issuer originale.

- **Credential Issuer Diventa Inattivo**: Se un Credential Issuer diventa inattivo, la sua **relativa** Entity Configuration e l'Endpoint di Entità Storica di Federazione DEVONO essere mantenuti disponibili.

6.11.1 Meccanismi di Attestazione di Trust Offline

I flussi offline non consentono la valutazione in tempo reale dello stato di un'Entità, come la sua revoca. Allo stesso tempo, l'utilizzo di Trust Chain di breve durata consente il raggiungimento di attestazioni di trust compatibili con i protocolli amministrativi di revoca richiesti (ad esempio, una revoca deve essere propagata in meno di 24 ore, quindi la Trust Chain non deve essere valida per più di quel periodo).

Attestazione di Trust del Wallet Offline

Dato che l'Istanza del Wallet non può pubblicare i suoi metadati online all'endpoint *.well-known/openid-federation*, DEVE ottenere un Wallet Attestation emesso dal suo Fornitore di Wallet. Il Wallet Attestation DEVE contenere tutte le informazioni rilevanti riguardanti le capacità di sicurezza dell'Istanza del Wallet e la sua configurazione relativa al protocollo. DOVREBBE contenere la Trust Chain relativa al suo emittente (Fornitore di Wallet).

Metadati del Relying Party Offline

Poiché la Scoperta di Entità di Federazione è applicabile solo in scenari online, è possibile includere la Trust Chain nelle richieste di presentazione che il Relying Party può emettere per un'Istanza del Wallet.

Il Relying Party DEVE firmare la richiesta di presentazione, la richiesta DOVREBBE includere il claim *trust_chain* nei suoi parametri di header JWT, contenente la Trust Chain di Federazione relativa a se stesso.

L'Istanza del Wallet che verifica la richiesta emessa dal Relying Party DEVE utilizzare le chiavi pubbliche del Trust Anchor per validare l'intera Trust Chain relativa al Relying Party prima di attestare la sua affidabilità.

Inoltre, l'Istanza del Wallet applica la policy dei metadati, se presente.

6.11.2 Rinnovo Rapido della Trust Chain

Il metodo di rinnovo rapido della Trust Chain offre un modo semplificato per mantenere la validità di una catena di trust senza sottoporsi nuovamente al processo di scoperta completo. È particolarmente utile per aggiornare rapidamente le Trust Relationship quando si verificano cambiamenti minori o quando la Trust Chain è vicina alla scadenza ma la struttura generale della federazione non è cambiata significativamente.

Il processo di rinnovo rapido della Trust Chain è iniziato recuperando nuovamente l'Entity Configuration della foglia. Tuttavia, a differenza del processo di scoperta di federazione che può comportare il recupero delle Entity Configuration a partire dagli authority hint, il rinnovo rapido si concentra sull'ottenere direttamente i Subordinate Statement. Queste dichiarazioni sono richieste utilizzando il *source_endpoint* fornito al loro interno, che punta alla posizione dove le dichiarazioni possono essere recuperate.

6.12 Meccanismo di Trust Evaluation

I Trust Anchor DEVONO distribuire le loro Chiavi Pubbliche di Federazione attraverso meccanismi sicuri fuori banda, come pubblicarle su una pagina web verificata o archiviarle in un repository remoto come parte di una lista di trust. La logica dietro questo requisito è che fare affidamento solo sui dati forniti all'interno dell'Entity Configuration del Trust Anchor non mitiga adeguatamente i rischi associati agli attacchi di manipolazione DNS e TLS. Per garantire la sicurezza, tutti i partecipanti DEVONO ottenere le chiavi pubbliche del Trust Anchor utilizzando questi metodi fuori banda. Dovrebbero quindi confrontare queste chiavi con quelle ottenute dall'Entity Configuration del Trust Anchor, scartando qualsiasi chiave che non corrisponda. Questo processo aiuta a garantire l'integrità e l'autenticità delle chiavi pubbliche del Trust Anchor e la sicurezza generale della federazione.

Il Trust Anchor pubblica l'elenco dei suoi Subordinati (endpoint di Elenco Subordinati di Federazione) e le attestazioni dei loro metadati e chiavi pubbliche (Subordinate Statement).

Ogni partecipante, inclusi Trust Anchor, Intermediario, Credential Issuer, Fornitore di Wallet e Relying Party, pubblica i propri metadati e chiavi pubbliche (endpoint Entity Configuration) nella risorsa web well-known **.well-known/openid-federation**.

Ognuno di questi può essere verificato utilizzando il Subordinate Statement emesso da un superiore, come il Trust Anchor o un Intermediario.

Ogni Subordinate Statement è verificabile nel tempo e DEVE avere una data di scadenza. La revoca di ogni dichiarazione è verificabile in tempo reale e online (solo per flussi remoti) attraverso gli endpoint di federazione.

Nota

La revoca di un'Entità è fatta con l'indisponibilità del Subordinate Statement relativo ad essa. Se il Trust Anchor o il suo Intermediario non pubblica un Subordinate Statement valido, o se pubblica un Subordinate Statement scaduto/non valido, il soggetto del Subordinate Statement DEVE essere inteso come non valido o revocato.

La concatenazione delle dichiarazioni, attraverso la combinazione di questi meccanismi di firma e il binding di claim e chiavi pubbliche, forma la Trust Chain.

Le Trust Chain possono anche essere verificate offline, utilizzando una delle chiavi pubbliche del Trust Anchor.

Nota

Poiché l'Istanza del Wallet non è un'Entità di Federazione, il Meccanismo di Trust Evaluation relativo ad essa **richiede la presentazione del Wallet Attestation durante le fasi di emissione e presentazione delle credenziali**.

Il Wallet Attestation trasmette tutte le informazioni richieste riguardanti l'istanza, come la sua chiave pubblica e qualsiasi altra informazione tecnica o amministrativa, senza alcun dato personale dell'Utente.

Ruolo nell'Onboarding: I meccanismi di trust evaluation sono essenziali durante la registrazione delle entità e la gestione del ciclo di vita per verificare lo stato di conformità dei nuovi partecipanti e validare i cambiamenti alle entità esistenti. Questo include la validazione delle credenziali di registrazione delle entità che richiedono l'onboarding e garantire che soddisfino le policy di federazione prima di concedere l'autorizzazione operativa.

Ruolo nelle Operazioni: Durante le operazioni di emissione e presentazione delle credenziali, la trust evaluation fornisce verifica in tempo reale della validità dell'entità e dello stato di conformità. Questo abilita transazioni di credenziali sicure garantendo che tutte le entità partecipanti (credential issuer, relying party, fornitori di wallet) mantengano il loro stato autorizzato e siano conformi alle policy di federazione correnti.

6.12.1 Stabilire Trust con i Credential Issuer

Nel processo di emissione, la Trust Evaluation garantisce l'integrità e l'autenticità delle Credenziali emesse e l'affidabilità dei loro Emittenti. Questa sezione delinea i meccanismi di Trust Evaluation distinti dai flussi di protocollo, implementati dalle Istanze del Wallet e dai Relying Party, come descritto nella sezione dedicata.

Le Trust Evaluation implementano modi diversi, come definito di seguito:

- **Scoperta di Entità di Federazione**: Le Istanze del Wallet e i Relying Party DEVONO verificare l'identità dell'Emittente utilizzando il processo di Scoperta di Entità di Federazione definito in *Discovery della Federazione* (pagina 135). Questo comporta l'interrogazione degli endpoint di federazione per confermare lo stato di validità dell'Emittente e la conformità al Trust Framework.
- **Trust Chain**: Le Istanze del Wallet e i Relying Party valutano le Trust Chain dell'Emittente utilizzando i meccanismi definiti in *Trust Chain* (pagina 136). Le Trust Chain possono essere fornite staticamente o costruite attraverso un processo di Scoperta di Entità di Federazione, per garantire che l'entità che richiede la Credenziale faccia parte di una federazione riconosciuta e fidata.
- **Valutazione dei Trust Mark**: I Trust Mark sono valutati per garantire la conformità continua alle policy di federazione. Questi marchi indicano l'aderenza a standard e pratiche specifici richiesti dalla federazione.
- **Valutazione delle Policy**: Le Istanze del Wallet e i Relying Party DEVONO verificare che il Credential Issuer sia autorizzato nell'emissione della Credenziale di loro interesse. Metadati, policy dei metadati e Trust Mark sono utilizzati per l'implementazione di questi controlli (*WP_050a* (pagina 790)).

Nel processo rappresentato nel diagramma di sequenza sottostante, l'Istanza del Wallet utilizza l'API di Federazione per scoprire e raccogliere tutti i Credential Issuer abilitati all'interno della federazione. Il processo di scoperta produce la Trust Chain. Quando la Trust Chain è fornita staticamente all'interno di una richiesta firmata o Credenziale, RICHIEDE solo di essere aggiornata quando la connessione internet è disponibile, mentre DEVE essere aggiornata quando la Trust Chain fornita staticamente risulta scaduta.

Nota

Come mostrato nella figura, il processo di Trust Evaluation è completamente separato e distinto dal flusso specifico del protocollo. Opera in un flusso diverso e utilizza protocolli specializzati progettati specificamente per questo scopo.

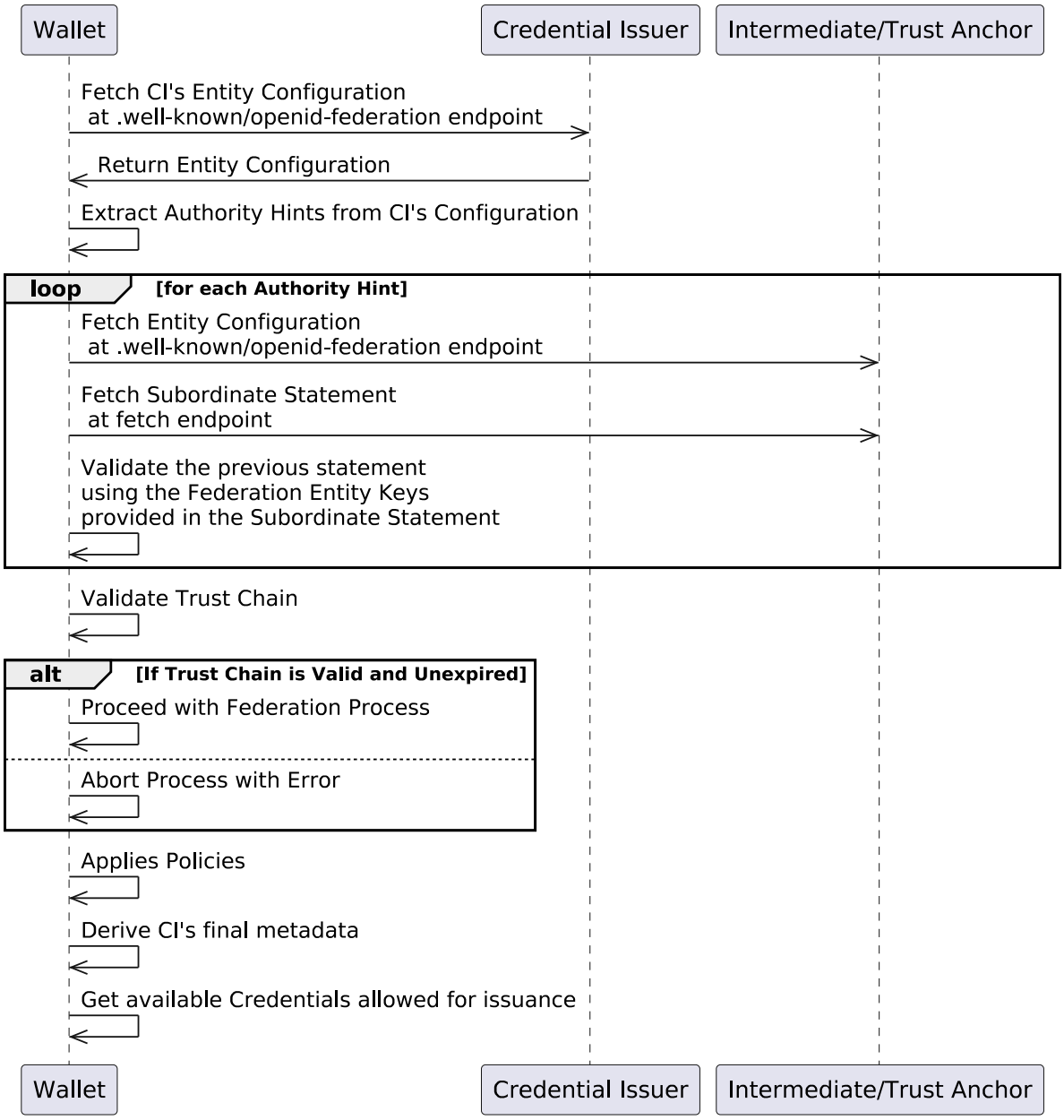


Fig. 6.2: Processo di Trust Evaluation.

6.12.2 Stabilire Trust con il Relying Party

Nel contesto della valutazione dei Relying Party, la responsabilità per la Trust Evaluation spetta esclusivamente all'Istanza del Wallet. I meccanismi di Trust Evaluation sono distinti dai flussi di protocollo e sono implementati dall'Istanza del Wallet, come dettagliato nella sezione dedicata.

Le Trust Evaluation sono condotte come segue:

- **Scoperta di Entità di Federazione**: Quando l'Istanza del Wallet riceve una richiesta firmata emessa da un Relying Party, l'Istanza del Wallet DEVE verificare l'identità del Relying Party utilizzando il processo di Scoperta di Entità di Federazione definito in *Discovery della Federazione* (pagina 135). Questo comporta l'interrogazione degli endpoint di federazione per confermare lo stato di validità del Relying Party e la conformità al Trust Framework e valutare la firma della richiesta utilizzando il materiale crittografico ottenuto dalla Trust Chain.
- **Trust Chain**: L'Istanza del Wallet valuta le Trust Chain del Relying Party utilizzando i meccanismi definiti in *Trust Chain* (pagina 136). Le Trust Chain possono essere fornite staticamente o costruite attraverso un processo di Scoperta di Entità di Federazione, per garantire che il Relying Party faccia parte di una federazione riconosciuta e fidata (*WP_079* (pagina 798)).
- **Valutazione dei Trust Mark**: I Trust Mark sono valutati per garantire la conformità continua alle policy di federazione. Questi marchi indicano l'aderenza a standard e pratiche specifici richiesti dalla federazione. I Relying Party POSSONO includere Trust Mark nella loro Entity Configuration per segnalare proprietà amministrative e conformità a profili specifici, come le concessioni nell'interagire con utenti minorenni (*WP_080* (pagina 798)).
- **Valutazione delle Policy**: L'Istanza del Wallet DEVE verificare che il Relying Party sia autorizzato a richiedere la Credenziale di interesse. Metadati, policy dei metadati e Trust Mark sono utilizzati per implementare questi controlli (*WP_087* (pagina 798)).

Nel processo raffigurato nel diagramma di sequenza sottostante, l'Istanza del Wallet utilizza l'API di Federazione per scoprire e raccogliere tutti i Relying Party abilitati all'interno della federazione. Il processo di scoperta produce la Trust Chain. Quando la Trust Chain è fornita staticamente all'interno di una richiesta firmata, deve solo essere aggiornata quando una connessione internet è disponibile, ma DEVE essere aggiornata se la Trust Chain fornita staticamente è scaduta.

Nota

Come mostrato nella figura, è richiesta una connessione internet per aggiornare la Trust Chain su un RP e controllare il suo stato di revoca.

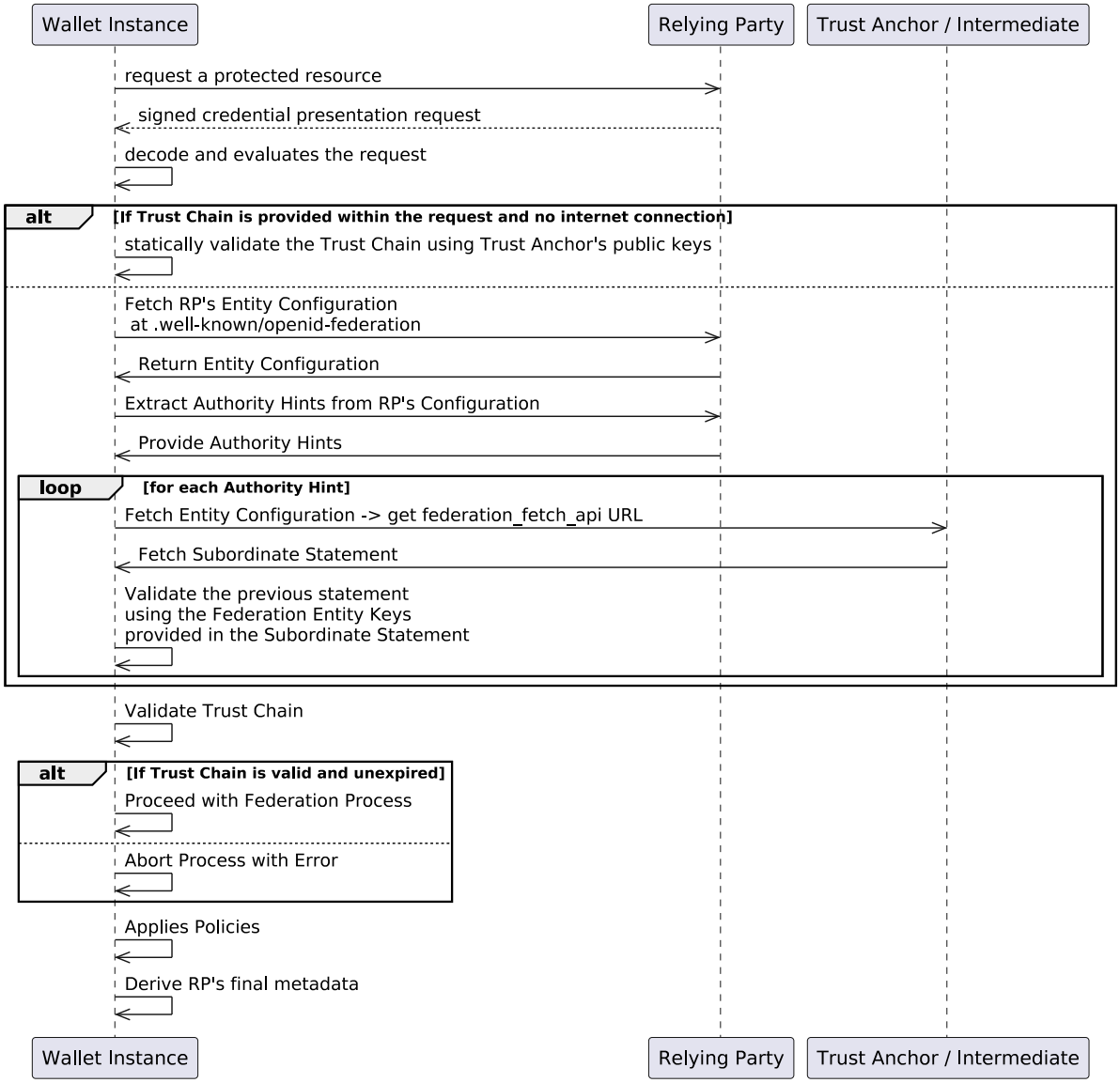


Fig. 6.3: Trust Evaluation - Processo di Scoperta del Relying Party.

Stabilire Trust con il Relying Party che opera tramite Intermediario

Quando un Relying Party opera tramite un Intermediario di Relying Party, l'Istanza del Wallet DEVE eseguire un processo di Trust Evaluation esteso che include la validazione del Trust Mark dell'Intermediario di Relying Party. Questo processo si svolge durante la fase di costruzione e validazione della Trust Chain e consente al Wallet di presentare all'Utente informazioni trasparenti sull'entità che ha registrato e garantisce il RP.

Il processo si articola nei seguenti passi:

1. **Download dell'Entity Configuration del RP**: L'Istanza del Wallet scarica l'Entity Configuration del Relying Party dall'endpoint `/.well-known/openid-federation` del RP. L'EC del RP contiene il claim `authority_hints` che punta all'URL dell'Intermediario di Relying Party (non direttamente al Trust Anchor).
2. **Identificazione dell'Intermediario di Relying Party**: Tramite il claim `authority_hints` nell'EC del RP, l'Istanza del Wallet identifica l'Intermediario di Relying Party e ne scarica l'Entity Configuration.
3. **Validazione del Trust Mark dell'Intermediario di Relying Party**: L'Istanza del Wallet verifica che l'Entity Configuration dell'Intermediario di Relying Party contenga un Trust Mark valido per il ruolo di Intermediario di Relying Party, emesso dal Trust Anchor. La validità del Trust Mark DEVE essere verificata tramite l'endpoint `/trust_mark_status` del Trust Anchor.
4. **Costruzione della Trust Chain tramite l'Intermediario di Relying Party**: L'Istanza del Wallet costruisce la Trust Chain completa: Entity Configuration del RP → Subordinate Statement emesso dall'Intermediario di Relying Party per il RP → Subordinate Statement emesso dal Trust Anchor per l'Intermediario di Relying Party → Entity Configuration del Trust Anchor. Questa catena attesta che il RP è una Foglia riconosciuta dall'Intermediario di Relying Party e che l'Intermediario di Relying Party è a sua volta riconosciuto dal Trust Anchor.
5. **Presentazione all'Utente**: Una volta completata la validazione, l'Istanza del Wallet DEVE indicare all'Utente che il Relying Party richiedente opera tramite un Intermediario di Relying Party riconosciuto, mostrando le informazioni identificative dell'Intermediario di Relying Party (ad esempio `organization_name`) ottenute dalla sua Entity Configuration.

Nota

La presenza del Trust Mark con identificativo che termina con `federation-entity/openid_credential_verifier_intermediary` nell'EC dell'Intermediario di Relying Party costituisce la prova crittograficamente verificabile che l'entità è stata riconosciuta dal Trust Anchor come Intermediario di Relying Party autorizzato. L'Istanza del Wallet DEVE rifiutare la richiesta

del RP se la Trust Chain non può essere costruita e validata tramite un Intermediario di Relying Party riconosciuto o direttamente tramite il Trust Anchor.

6.12.3 Valutare Trust con i Wallet

Il Fornitore di Wallet emette il Wallet Attestation, certificando lo stato operativo delle sue Istanze del Wallet e includendo una delle loro chiavi pubbliche.

Il Wallet Attestation PUÒ contenere la Trust Chain che attesta l'affidabilità per il suo emittente (Fornitore di Wallet) al momento dell'emissione.

L'Istanza del Wallet fornisce il suo Wallet Attestation all'interno della richiesta firmata durante la fase di emissione PID/IT-Wallet ID. Il Credential Issuer DEVE valutare la Trust Chain sull'emittente del Wallet Attestation (formalmente, il Fornitore di Wallet).

6.13 Non-ripudio delle Attestazioni di Lunga Durata

Il Trust Anchor e i suoi Intermediari DEVONO esporre l'endpoint delle Chiavi Storiche di Federazione, dove sono pubblicate tutte le parti pubbliche delle Chiavi dell'Entità di Federazione che non sono più utilizzate, sia scadute che revocate.

I dettagli di questo endpoint sono definiti nella OID-FED Sezione 8.7.

Ogni JWT contenente una Trust Chain negli header JWT può essere verificato nel tempo, poiché l'intera Trust Chain è verificabile utilizzando la chiave pubblica del Trust Anchor.

Anche se il Trust Anchor ha cambiato le sue chiavi crittografiche per la firma digitale, l'endpoint delle Chiavi Storiche di Federazione rende sempre disponibili le chiavi non più utilizzate per le verifiche di firma storiche.

6.14 X.509 PKI

L'Infrastruttura a Chiave Pubblica (PKI) X.509 è un framework progettato per creare, gestire, distribuire, utilizzare, archiviare e revocare certificati digitali X.509. Al centro della PKI X.509 c'è il concetto di Autorità di Certificazione (CA), che emette certificati digitali alle entità. Questi certificati sono richiesti per stabilire comunicazioni sicure su reti, incluso internet, abilitando funzionalità di crittografia e firma digitale. La gerarchia PKI tipicamente coinvolge una CA radice in cima, con una o più CA subordinate sotto, formando una catena di trust. Le entità si affidano a questa catena di trust per verificare l'autenticità dei certificati. Gli standard X.509 definiscono il formato dei certificati a chiave pubblica.

L'integrazione di OpenID Federation 1.0 con la PKI tradizionale basata su X.509 (rfc:5280), completata da un'API RESTful, mira a migliorare l'infrastruttura con funzionalità aggiuntive, rendendola navigabile e trasparente.

Questo approccio sfrutta la natura dinamica e flessibile di OpenID Federation insieme al requisito dei Certificati X.509 per applicazioni legacy e scopi di interoperabilità, mirando ad affrontare le esigenze in evoluzione di verifica dello stato di registrazione dei partecipanti alla federazione, la loro conformità alle regole condivise e la gestione generale e interoperabile del trust in ecosistemi digitali multilaterali.

Ruolo nell'Onboarding: Durante la registrazione delle entità, i certificati X.509 completano i meccanismi di OpenID Federation fornendo interoperabilità con sistemi legacy e abilitando l'integrazione con infrastrutture PKI esistenti. Le entità auto-emettono certificati X.509 utilizzando le loro chiavi di federazione, estendendo le relazioni di trust ai sistemi tradizionali basati su certificati.

Ruolo nelle Operazioni: Durante le operazioni di credenziali, i certificati X.509 abilitano comunicazioni sicure con sistemi legacy e forniscono percorsi di verifica alternativi per entità che richiedono validazione PKI tradizionale. Questo approccio duale garantisce che l'infrastruttura IT-Wallet possa interoperare con sistemi legacy esistenti mantenendo meccanismi di trust moderni basati su federazione.

OpenID Federation e PKI basata su X.509 condividono diverse cose in comune, come elencato di seguito:

- **Approccio Gerarchico**: entrambi utilizzano un Trust Model gerarchico con una singola terza parte fidata sovrastante, nota come Trust Anchor, che è fidata sopra tutte le altre.
- **Decentralizzazione con Multipli Trust Anchor e Intermediari**: nonostante un modello gerarchico unico, la possibilità di avere multipli Trust Anchor e Intermediari, sotto uno o più Trust Anchor, introduce un livello di decentralizzazione.
- **Estensioni Personalizzate**: entrambi i sistemi consentono estensioni personalizzate per soddisfare requisiti specifici o per migliorare la funzionalità. I Certificati X.509 supportano estensioni personalizzate, OpenID Federation consente la definizione di metadati specifici del protocollo personalizzati, Trust Mark e policy utilizzando un Policy Language.
- **Catena di Trust/Certificato**: si affidano a una prova concatenata di trust, dove il trust è passato dall'autorità radice (Trust Anchor) attraverso Intermediari all'entità finale (Foglia).
- **Vincoli nella Catena**: i vincoli possono essere applicati all'interno della Trust Chain riguardo aspetti critici come la delegazione del trust, il numero di intermediari e i domini coinvolti.

- **Distribuzione di Chiavi Pubbliche**: Entrambi i sistemi coinvolgono la distribuzione della chiave pubblica del Trust Anchor per garantire che le entità possano verificare la catena di trust.
- **Registro di Chiavi Scadute**: Mantenere un registro di chiavi scadute è cruciale per entrambi, garantendo il non-ripudio delle firme passate anche quando le chiavi cambiano.

6.14.1 Trust Anchor di Federazione e CA X.509

Nel contesto di OpenID Federation, il Trust Anchor gioca un ruolo simile a quello di un'Autorità di Certificazione (CA) nelle Infrastrutture a Chiave Pubblica (PKI) basate su X.509. Entrambi servono come elementi fondamentali di trust all'interno dei loro rispettivi sistemi. In questo documento, il termine "Trust Anchor" è spesso utilizzato per comprendere entrambi i concetti. L'infrastruttura di trust descritta qui allinea il Trust Anchor di OpenID Federation con l'Autorità di Certificazione PKI X.509, rendendoli quindi un'unica entità unica che supporta sia RFC 5280 che OpenID Federation 1.0.

6.14.2 Emissione di Certificati X.509

In una Federazione OpenID, ogni partecipante è richiesto di auto-emettere la sua Entity Configuration, firmandola con una delle sue chiavi crittografiche che sono attestate dai Superiori Immediati.

Allo stesso modo, ogni Entità di federazione ha l'autonomia di emettere una dichiarazione firmata su se stessa sotto forma di un Certificato X.509. I partecipanti alla federazione che hanno bisogno di emettere Certificati X.509 su se stessi e per i loro scopi specifici, possono emettere e firmare Certificati X.509 utilizzando una delle loro Chiavi dell'Entità di Federazione attestate dalle loro Autorità di Federazione (Superiore Immediato). Questo processo allinea l'emissione di Certificati X.509 con il paradigma di delegazione della federazione.

Questo è fattibile perché il Certificato X.509 può essere verificato utilizzando una Catena di Certificati X.509, simile all'approccio utilizzato per le Entity Configuration in OpenID Federation.

Le Foglie di Federazione non sono Autorità di Certificazione (CA) o intermediari CA autorizzati a emettere certificati X.509 per i loro subordinati. Invece, le Foglie di Federazione agiscono come intermediari per l'emissione di certificati esclusivamente su se stesse. Questo è realizzato applicando vincoli di denominazione appropriati per garantire che i certificati X.509 siano correttamente delimitati. I vincoli di denominazione sono applicati dai Superiori Immediati all'interno dei certificati emessi all'entità Foglia, specificamente riguardo alle Chiavi dell'Entità di Federazione della Foglia. Di conseguenza, la Foglia può emettere certificati X.509 solo su se stessa, mantenendo così l'integrità della Trust Chain.

Quando un partecipante auto-emette un Certificato X.509, aderisce ai seguenti requisiti:

1. **Nome del Soggetto**: Il nome del soggetto del Certificato X.509 DEVE corrispondere all'identità del partecipante. Il nome del soggetto degli Intermediari e delle Foglie DEVE includere i seguenti attributi:

Attributo	Requisito
`Country Name (C)`	DEVE contenere il codice paese ISO a due lettere.
`State or Province Name (ST)`	DEVE contenere la regione o stato dove l'entità è localizzata.
`Locality Name (L)`	DEVE contenere la città dove l'entità è localizzata.
`Organization Name (O)`	DEVE contenere il nome legale dell'organizzazione.
`Organizational Unit Name (OU)`	PUÒ contenere il nome del dipartimento all'interno dell'organizzazione (opzionale).
`Common Name (CN)`	DEVE contenere il nome DNS dell'identificatore unico dell'Entità di Federazione, che è incluso nel valore `sub` (soggetto) nella sua Entity Configuration di federazione, rimuovendo `https://` e qualsiasi path web.
`Email Address`	DEVE contenere l'indirizzo email di contatto dell'organizzazione.
`organizationIdentifier`	DEVE contenere il numero di registrazione che identifica univocamente l'organizzazione all'interno del servizio di registrazione, utilizzando il valore OID `2.5.4.97` come definito in `ITU-T X.500`.

2. **Subject Alternative Name (SAN)**: Il Certificato X.509 DEVE includere un `SAN URI` che DEVE corrispondere ai valori **sub** e **iss** della sua Entity Configuration di federazione.

3. **Nome DNS**: Il Certificato X.509 DEVE includere un Nome DNS nel SAN che corrisponde al nome DNS contenuto all'interno dei valori **sub** e **iss** della sua Entity Configuration, rimuovendo `https://` e qualsiasi path web.

4. **Certificate Revocation List (CRL)**: Se il Certificato X.509 emesso ha un tempo di scadenza superiore a 24 ore, l'Emittente X.509 DEVE pubblicare una CRL per i Certificati X.509 emessi. Questo elenco DEVE essere accessibile e regolarmente aggiornato per garantire che qualsiasi Certificato X.509 compromesso o non valido sia prontamente revocato con la motivazione della revoca, se presente.

5. **Basic Constraints**: Il Certificato X.509 DEVE includere un'estensione `Basic Constraints` con `CA:TRUE` e una lunghezza massima del path di 1 se l'emittente del certificato è un Intermediario di Federazione. Se è una Foglia, la lunghezza massima del path DEVE essere impostata a 0. Questo indica che il Subordinato a cui il certificato si riferisce, può emettere Certificati X.509 solo su se stesso. L'estensione `BasicConstraints` DEVE essere impostata `critical`.

6. **Key Usage**: `Digital Signature`, `Key Encipherment`, `Certificate Sign`, `CRL Sign` DEVONO essere inclusi. L'estensione `KeyUsage` DEVE essere impostata `critical`.

7. **Name Constraints**: Il Certificato X.509 DEVE includere `Name Constraints` per specificare domini e URI permessi ed esclusi. Per esempio:
   - Permessi: - `URI.1=https://leaf.example.com` - `DNS.1=leaf.example.com`
   - Esclusi: - `DNS=localhost` - `DNS=localhost.localdomain` - `DNS=127.0.0.1` - `DNS=example.com` - `DNS=example.org` - `DNS=example.net` - `DNS=*.example.org`
8. **AuthorityKeyIdentifier**: Il Certificato X.509 DEVE includere un'estensione `Authority KeyIdentifier`. Il campo `keyIdentifier` dell'estensione `AuthorityKeyIdentifier` DEVE essere presente e DEVE essere identico al campo `SubjectKeyIdentifier` del certificato dell'emittente. Questo consolida la costruzione e validazione della catena di certificati.

Di seguito è riportato un esempio non normativo, in testo semplice (formato OpenSSL), di una catena di certificati X.509 con una CA intermedia, a partire dal certificato Foglia.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 950713202307300703483739727303172061473531557 2
        (0x1aa5085f6cfe919f8f4ae34a8ca97e2da1f0274)
    Signature Algorithm: sha256
        Issuer: commonName=intermediate.example.org, countryName=IT,
        emailAddress=info@intermediate.example.org, organizationName=Example Intermediate
        Validity
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=leaf.example.it, countryName=IT,
        emailAddress=info@leaf.example.it, organizationName=Example Leaf
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X: 62908053918297774357647421014545143197415914813624128496938845341292839993 55
                Y:
                112976873315937677211730342235778677869832940825390486712747813176515478885564
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:1
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://leaf.example.it/crl/leaf.example.it.crl
            X509v3 Name Constraints:
                Permitted:
                  DNS.1=leaf.example.it
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
```

```
                IP.3=127.0.0.1/32
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:https://leaf.example.it/openid4vp
                DNS:leaf.example.it
            X509v3 Subject Key Identifier:
                10:45:d4:c1:22:e0:f1:c2:1f:a2:84:42:df:71:0f:ba:88:16:e0:a9
            X509v3 Authority Key Identifier:
                keyid:07:4b:be:d2:bf:57:db:ce:40:e4:fa:c2:89:27:db:ff:84:8f:7c:7c

    Signature Algorithm: sha256
     30:44:02:20:64:3f:1f:a5:15:b0:62:0c:1b:c2:a3:06
     7f:e0:16:0c:cb:a4:e5:7b:70:e3:31:3a:67:5a:a9:80
     24:23:af:f1:02:20:03:fe:75:fa:75:66:1f:1c:84:30
     c3:74:57:c7:25:3b:58:05:bf:38:b9:0f:b7:88:cb:9f
     3a:6e:0c:60:61:8b

-----BEGIN CERTIFICATE-----
MIIDIzCCAsqgAwIBAgIUAapQhfbP6Rn49K40qMqX4tofAnQwCgYIKoZIzj0EAwIw
fTEhMB8GA1UEAwwYaW50ZXJtZWRpYXRlLmV4YW1wbGUub3JnMQswCQYDVQQGEwJJ
VDEsMCoGCSqGSIb3DQEJARYdaW5mb0BpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcx
HTAbBgNVBAoMFEV4YW1wbGUgSW50ZXJtZWRpYXRlMB4XDTI1MDkyODEwMTQzOFoX
DTI2MDkyOTEwMTQzOFowYzEYMBYGA1UEAwwPbGVhZi5leGFtcGxlLml0MQswCQYD
VQQGEwJJVDEjMCEGCSqGSIb3DQEJARYUaW5mb0BsZWFmLmV4YW1wbGUuaXQxFTAT
BgNVBAoMDEV4YW1wbGUgTGVhZjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABA3o
eDyFgrsWCLfLka1OAaQ5KGFyB0gZuLFMcT2GAqZ7+cak7eOb0wrX9c+n6vhlkg8D
hTD5luafdkOZUXSbFLyjggFAMIIBPDASBgNVHRMBAf8ECDAGAQH/AgEBMEAGA1Ud
HwQ5MDcwNaAzoDGGL2h0dHBzOi8vbGVhZi5leGFtcGxlLml0L2NybC9sZWFmLmV4
YW1wbGUuaXQuY3JsMFUGA1UdHgEB/wRLMEmgEzARgg9sZWFmLmV4YW1wbGUuaXSh
MjALgglsb2NhbGhvc3QwF4IVbG9jYWxob3N0LmxvY2FsZG9tYWluMAqHCH8AAAH/
////MA4GA1UdDwEB/wQEAwIBpjA9BgNVHREENjA0hiFodHRwczovL2xlYWYuZXhh
bXBsZS5pdC9vcGVuaWQ0dnCCD2xlYWYuZXhhbXBsZS5pdDAdBgNVHQ4EFgQUEEXU
wSLg8cIfooRC33EPuogW4KkwHwYDVR0jBBgwFoAUB0u+0r9X285A5PrCiSfb/4SP
fHwwCgYIKoZIzj0EAwIDRwAwRAIgZD8fpRWwYgwbwqMGf+AWDMuk5Xtw4zE6Z1qp
gCQjr/ECIAP+dfp1Zh8chDDDdFfHJTtYBb84uQ+3iMufOm4MYGGL
-----END CERTIFICATE-----

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 148177522658585329990858238295912831877447595707
        (0x19f4828a0290c333d92efe8d9c5306b979e6b6bb)
    Signature Algorithm: sha256
        Issuer: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Validity
```

```
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=intermediate.example.org, countryName=IT,
        emailAddress=info@intermediate.example.org, organizationName=Example Intermediate
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X: 98817418608085252616249916893677319857297304171504928731420155197485062951 27
                Y:
                100724824207124402297793814607603963819791163379056466342838110166095246764680
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:2
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://intermediate.example.org/crl/intermediate.example.org.crl
            X509v3 Name Constraints:
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
                  IP.3=127.0.0.1/32
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:https://intermediate.example.org
                DNS:intermediate.example.org
            X509v3 Subject Key Identifier:
                07:4b:be:d2:bf:57:db:ce:40:e4:fa:c2:89:27:db:ff:84:8f:7c:7c
            X509v3 Authority Key Identifier:
                keyid:95:b9:12:12:d0:c2:48:cc:20:2d:6c:54:96:f3:a2:7c:72:68:5c:31

    Signature Algorithm: sha256
     30:45:02:21:00:c2:98:24:30:7e:4a:69:bc:d2:b8:e0
     1c:aa:c5:98:34:57:63:d3:7f:3f:dc:4e:53:30:6e:4d
     c5:6d:64:2e:b2:02:20:2a:08:81:86:a4:06:c5:7f:84
     ab:7b:9e:1c:42:b7:17:84:c7:9c:04:a4:8a:84:22:50
     3d:71:c2:e9:10:28:54
```

```
-----BEGIN CERTIFICATE-----
MIIDJTCCAsugAwIBAgIUGfSCigKQwzPZLv6NnFMGuXnmtrswCgYIKoZIzj0EAwIw
XzEXMBUGA1UEAwwOY2EuZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZI
hvcNAQkBFhNpbmZvQGNhLmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENB
MB4XDTI1MDkyODEwMTQzOFoXDTI2MDkyOTEwMTQzOFowfTEhMB8GA1UEAwwYaW50
ZXJtZWRpYXRlLmV4YW1wbGUub3JnMQswCQYDVQQGEwJJVDEsMCoGCSqGSIb3DQEJ
ARYdaW5mb0BpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcxHTAbBgNVBAoMFEV4YW1w
bGUgSW50ZXJtZWRpYXRlMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEFdjeIOVq
```

```
+Ah2AagzWx69sHvjV2K7OkBPTyxnE63Qm1fesDruwn8LexbY0Y9qfdhpyMPaSR+A
3R988KLkypGqiKOCAUUwggFBMBIGA1UdEwEB/wQIMAYBAf8CAQIwUgYDVR0fBEsw
STBHoEWgQ4ZBaHR0cHM6Ly9pbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcvY3JsL2lu
dGVybWVkaWF0ZS5leGFtcGxlLm9yZy5jcmwwQAYDVR0eAQH/BDYwNKEyMAuCCWxv
Y2FsaG9zdDAXghVsb2NhbGhvc3QubG9jYWxkb21haW4wCocIfwAAAf////8wDgYD
VR0PAQH/BAQDAgGmMEUGA1UdEQQ+MDyGIGh0dHBzOi8vaW50ZXJtZWRpYXRlLmV4
YW1wbGUub3JnghhpbnRlcm1lZGlhdGUuZXhhbXBsZS5vcmcwHQYDVR0OBBYEFAdL
vtK/V9vOQOT6wokn2/+Ej3x8MB8GA1UdIwQYMBaAFJW5EhLQwkjMIC1sVJbzonxy
aFwxMAoGCCqGSM49BAMCA0gAMEUCIQDCmCQwfkppvNK44ByqxZg0V2PTfz/cTlMw
bk3FbWQusgIgKgiBhqQGxX+Eq3ueHEK3F4THnASkioQiUD1xwukQKFQ=
-----END CERTIFICATE-----

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 660157339746673467767137473267076395071677825012
        (0x73a27a9ad0b1a3c0915c78404a7ea644a1de7ff4)
    Signature Algorithm: sha256
        Issuer: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Validity
            Not Before: Sep 28 10:14:38 2025 GMT
            Not After : Sep 29 10:14:38 2026 GMT
        Subject: commonName=ca.example.com, countryName=IT, emailAddress=info@ca.example.com,
        organizationName=Example CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey (EC)
                Public-Key: (256 bit)
                Curve: secp256r1
                X:
                60510079538073841265704016785331040638654008648926352025877164647615009972064
                Y:
                94446385337085233641787598544598093044189082897506261545395904518792084782698
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://ca.example.com/crl/ca.example.com.crl
            X509v3 Name Constraints:
                Excluded:
                  DNS.1=localhost
                  DNS.2=localhost.localdomain
                  IP.3=127.0.0.1/32
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
```

```
                URI:https://ca.example.com
                DNS:ca.example.com
            X509v3 Subject Key Identifier:
                95:b9:12:12:d0:c2:48:cc:20:2d:6c:54:96:f3:a2:7c:72:68:5c:31

    Signature Algorithm: sha256
     30:45:02:21:00:c5:98:db:a0:5a:cd:d2:fe:a7:f3:c2
     12:b4:f5:2f:14:d2:a7:cc:17:ea:5e:ce:b2:69:7e:a9
     0f:5a:95:ab:a6:02:20:6d:b7:9b:e0:22:01:3e:e0:a0
     ac:2a:51:53:e8:73:e3:0d:b4:fb:fb:af:dc:91:56:3b
     14:c2:31:c9:d4:41:49

-----BEGIN CERTIFICATE-----
MIICuTCCAl+gAwIBAgIUc6J6mtCxo8CRXHhASn6mRKHef/QwCgYIKoZIzj0EAwIw
XzEXMBUGA1UEAwwOY2EuZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZI
hvcNAQkBFhNpbmZvQGNhLmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENB
MB4XDTI1MDkyODEwMTQzOFoXDTI2MDkyOTEwMTQzOFowXzEXMBUGA1UEAwwOY2Eu
ZXhhbXBsZS5jb20xCzAJBgNVBAYTAklUMSIwIAYJKoZIhvcNAQkBFhNpbmZvQGNh
LmV4YW1wbGUuY29tMRMwEQYDVQQKDApFeGFtcGxlIENBMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEhcd9yAMYDQ3pJ6mQy8nFpTOTS5khBJ35dLryF+z832DQzsJ+
Bm2AgrtqOOioLRLrSz1UNSXkY08aVGr0vcz+aqOB+DCB9TAPBgNVHRMBAf8EBTAD
AQH/MD4GA1UdHwQ3MDUwM6AxoC+GLWh0dHBzOi8vY2EuZXhhbXBsZS5jb20vY3Js
L2NhLmV4YW1wbGUuY29tLmNybDBABgNVHR4BAf8ENjA0oTIwC4IJbG9jYWxob3N0
MBeCFWxvY2FsaG9zdC5sb2NhbGRvbWFpbjAKhwh/AAAB/////zAOBgNVHQ8BAf8E
BAMCAaYwMQYDVR0RBCowKIYWaHR0cHM6Ly9jYS5leGFtcGxlLmNvbYIOY2EuZXhh
bXBsZS5jb20wHQYDVR0OBBYEFJW5EhLQwkjMIC1sVJbzonxyaFwxMAoGCCqGSM49
BAMCA0gAMEUCIQDFmNugWs3S/qfzwhK09S8U0qfMF+pezrJpfqkPWpWrpgIgbbeb
4CIBPuCgrCpRU+hz4w20+/uv3JFWOxTCMcnUQUk=
-----END CERTIFICATE-----
```

Utilizzando il livello sottostante stabilito con OpenID Federation 1.0, tutti i certificati X.509 sono emessi in modo propriamente decentralizzato utilizzando il pattern di delegazione.

6.14.3 Revoca di Certificati X.509

Un Certificato X.509 può essere revocato dal suo Emittente. Le liste di revoca, e/o qualsiasi altro meccanismo di controllo della revoca, sono particolarmente richieste per i Certificati X.509 con un tempo di scadenza superiore a 24 ore; altrimenti, non sono richieste.

Quando l'emittente del Certificato X.509 è la Foglia e quindi il Certificato X.509 si riferisce a se stesso, se il tempo di scadenza del certificato è superiore a 24 ore dal tempo `X509_NOT_VALID_BEFORE`, DEVE implementare una CRL relativa al certificato emesso e mantenerla aggiornata. Quando l'emittente del Certificato X.509 è un Superiore immediato, come il Trust Anchor o un Intermediario, e revoca il certificato relativo alla Foglia, cioè il Certificato X.509 relativo a una delle Chiavi dell'Entità di Federazione della Foglia, questa azione invalida l'intera Trust

Chain associata a quella chiave pubblica crittografica della Foglia, rimuovendo effettivamente la sua capacità di emettere ulteriori Certificati X.509 su se stessa. Questo meccanismo di revoca gerarchico garantisce che qualsiasi compromissione o comportamento scorretto da parte di un'entità Foglia possa essere rapidamente affrontato.

Di seguito è riportato un esempio non normativo, in testo semplice, che illustra il contenuto di una CRL.

```
Certificate Revocation List (CRL):
Version: 2 (0x1)
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=leaf.example.org, O=Leaf, C=IT
Last Update: Sep 1 00:00:00 2023 GMT
Next Update: Sep 8 00:00:00 2023 GMT
Revoked Certificates:
    Serial Number: 987654320
        Revocation Date: Aug 25 12:00:00 2023 GMT
        CRL Entry Extensions:
            Reason Code: Key Compromise
    Serial Number: 987654321
        Revocation Date: Aug 30 15:00:00 2023 GMT
        CRL Entry Extensions:
            Reason Code: Cessation of Operation
Signature Algorithm: sha256WithRSAEncryption
Signature:
    5c:4f:3b:...
```

6.14.4 Federation Subordinate Events Endpoint

L'Endpoint Eventi Subordinati della Federazione è definito in OID-FED-SUBORDINATE-EVENTS. Questo endpoint fornisce un meccanismo per Trust Anchor e Intermediari per pubblicare eventi storici relativi ai loro Subordinati Immediati. Fornisce trasparenza e responsabilità all'interno della federazione fornendo un record storico completo di eventi significativi che influenzano i partecipanti della federazione.

Per i dettagli completi della specifica, inclusi la posizione dell'endpoint, il formato della richiesta, il formato della risposta, i claim JWT, i parametri degli oggetti evento e i tipi di evento supportati, fare riferimento a OID-FED-SUBORDINATE-EVENTS.

Di seguito è riportato un esempio non normativo di una richiesta e risposta dell'Endpoint Eventi Subordinati della Federazione:

Esempio di Richiesta:

```
GET /federation_subordinate_events_endpoint?sub=https%3A%2F%2Frp%2Eexample%2Eorg HTTP/1.1
Host: immediate-superior.example.org
```

Esempio di Risposta:


```
{
  "iss": "https://immediate-superior.example.org",
  "sub": "https://rp.example.org",
  "iat": 1590000000,
  "federation_registration_events": [
    {
      "iat": 1590000000,
      "event": "registration"
    },
    {
      "iat": 1590000000,
      "event": "jwks_update"
    },
    {
      "iat": 1600000000,
      "event": "revocation",
      "event_description": "compromised node"
    },
    {
      "iat": 1610000000,
      "event": "registration"
    }
  ]
}
```


Integrazione con la Gestione del Ciclo di Vita delle Entità:

Questo endpoint completa le procedure di gestione del ciclo di vita delle entità definite in *Onboarding delle Entità* (pagina 239) fornendo un tracciamento dettagliato di tutti gli eventi significativi che influenzano i partecipanti della federazione. Supporta sia il monitoraggio automatizzato della conformità che i processi di audit manuale.

6.15 Osservazioni sulla Privacy

- Le Istanze del Wallet NON DEVONO pubblicare i loro metadati attraverso un servizio online.
- L'infrastruttura di trust DEVE essere pubblica, con tutti gli endpoint pubblicamente accessibili senza alcuna credenziale client che possa rivelare chi sta richiedendo l'accesso.
- Quando un'Istanza del Wallet richiede i Subordinate Statement per costruire la Trust Chain per un Relying Party specifico o valida un Trust Mark online, emesso per un Relying Party specifico, il Trust Anchor o il suo Intermediario non sanno che una particolare Istanza

del Wallet sta indagando su un Relying Party specifico; invece, servono solo le dichiarazioni relative a quel Relying Party come risorsa pubblica.

- I metadati dell'Istanza del Wallet NON DEVONO contenere informazioni che possano rivelare informazioni tecniche sull'hardware utilizzato.
- I metadati dell'entità Foglia, Intermediario e Trust Anchor possono includere la quantità necessaria di dati come parte delle informazioni di contatto amministrative, tecniche e di sicurezza. Generalmente non è raccomandato utilizzare dettagli di contatto personali in tali casi. Dal punto di vista legale, la pubblicazione di tali informazioni è necessaria per il supporto operativo riguardante questioni tecniche e di sicurezza e la regolamentazione GDPR.

6.16 Considerazioni sulla Decentralizzazione

- Ci può essere più di un singolo Trust Anchor.
- In alcuni casi, un verificatore di trust può fidarsi di un Intermediario, specialmente quando l'Intermediario agisce come Trust Anchor all'interno di un perimetro specifico, come casi dove le Foglie sono entrambe nello stesso perimetro come una giurisdizione di Stato Membro (ad esempio: un Relying Party italiano con un'Istanza del Wallet italiana può considerare l'Intermediario italiano come Trust Anchor per gli scopi delle loro interazioni).
- Le attestazioni di trust (Trust Chain) dovrebbero essere incluse nei JWT emessi dai Credential Issuer, e le Richieste di Presentazione degli RP dovrebbero contenere la Trust Chain relativa a loro (emittenti delle richieste di presentazione).
- Poiché la presentazione delle credenziali deve essere firmata, memorizzando le richieste e risposte di presentazione firmate, che includono la Trust Chain, l'Istanza del Wallet può avere lo snapshot della configurazione di federazione (Entity Configuration del Trust Anchor nella Trust Chain) e l'affidabilità verificabile del Relying Party con cui ha interagito.
- Ogni attestazione firmata è di lunga durata poiché può essere validata crittograficamente anche quando la configurazione di federazione cambia o le chiavi dei suoi emittenti sono rinnovate.
- Ogni partecipante dovrebbe essere in grado di aggiornare la sua Entity Configuration senza notificare i cambiamenti a qualsiasi terza parte. La policy dei metadati contenuta all'interno di una Trust Chain deve essere applicata per sovrascrivere qualsiasi informazione relativa ai metadati specifici del protocollo.

CAPITOLO 7

Infrastruttura del Registro

L'ecosistema IT-Wallet opera attraverso un'Infrastruttura del Registro che fornisce definizioni standardizzate dei dati, registrazione delle entità e capacità di discovery delle Credenziali. Il sistema di registro è composto da molteplici componenti interconnessi che supportano il ciclo di vita completo delle operazioni sulle Credenziali digitali, dall'onboarding delle entità alla presentazione delle Credenziali.

L'architettura del registro risponde ai requisiti di standardizzazione semantica, gestione della fiducia federata e discovery delle Credenziali attraverso componenti specializzati che garantiscono interoperabilità e conformità nell'intero ecosistema.

7.1 Panoramica dell'Architettura di Registro

Il Registro di Sistema IT-Wallet comprende sei componenti principali:

1. **Registro dei Claims**: Definizioni semantiche standardizzate per i singoli attributi delle Credenziali, i tipi di dati e le regole di validazione.
2. **Registro delle Fonti Autentiche (FA)**: Catalogo dei fornitori di dati registrati con le loro capacità dichiarate e i claim disponibili.
3. **Registro della Federazione**: Elenco autorevole delle entità fidate che partecipano alla federazione con le loro configurazioni tecniche.
4. **Catalogo delle Credenziali Digitali**: Meccanismo pubblico di discovery per le tipologie di Credenziale disponibili con i loro metadati e le informazioni di emissione.

5. **Registro degli Schemi**: Elenco autorevole degli Schemi di Credenziale.
6. **Tassonomia**: Sistema di classificazione gerarchica che organizza le Credenziali per dominio e finalità.

Questi componenti del registro sono interconnessi e mantenuti dall'Organismo di Vigilanza per garantire coerenza, sicurezza e conformità normativa nell'intero ecosistema.

7.2 Endpoint di Discovery del Registro

Il Trust Anchor DEVE fornire un meccanismo di discovery per tutti i componenti del registro attraverso endpoint *well-known* standardizzati che forniscono metadati e informazioni di discovery REST API per gestire operazioni complesse come paginazione e filtraggio.

Il Trust Anchor DEVE pubblicare i metadati di discovery del registro all'endpoint `.well-known/it-wallet-registry` con supporto alla negoziazione del contenuto:

- **Content-Type predefinito**: `application/jwt` (JWT firmato che garantisce autenticità e integrità)
- **Content-Type alternativo**: `application/json` (JSON semplice per scopi di sviluppo/debug)

Inoltre, il Registro di Sistema IT-Wallet DEVE utilizzare due distinti pattern di accesso:

- **API dei Registri Dati**: DEVONO supportare capacità di paginazione e filtraggio.
- **Infrastruttura di Fiducia della Federazione**: come definito in *L'Infrastruttura di Trust* (pagina 117).

Di seguito è riportato un esempio non normativo.

```
GET /.well-known/it-wallet-registry HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/jwt

HTTP/1.1 200 OK
Content-Type: application/jwt

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

```
GET /.well-known/it-wallet-registry HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
```

7.2.1 Parametri dell'Endpoint di Discovery del Registro

Il payload JWT della risposta dell'Endpoint di Discovery del Registro DEVE contenere i seguenti parametri:

Tabella 7.1: Endpoint di Discovery del Registro — Parametri del Payload JWT

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del documento di discovery (es. `urn:it-wallet-registry:it-wallet`).
version	OBBLIGATORIO. Versione del formato del documento di discovery (es. `1.0.0`).
last_modified	OBBLIGATORIO. Timestamp dell'ultima modifica al documento di discovery (es. `2024-03-15T10:30:00Z`).
endpoints	OBBLIGATORIO. Oggetto JSON contenente gli URI di tutti i componenti del registro. DEVONO essere presenti le seguenti chiavi di endpoint: • **claims_registry**: URI dell'API del Registro dei Claims. • **authentic_sources**: URI dell'API del Registro delle Fonti Autentiche. • **credential_catalog**: URI dell'endpoint well-known del Catalogo delle Credenziali Digitali. • **taxonomy**: URI della risorsa Tassonomia. • **schema_registry**: URI dell'API del Registro degli Schemi. • **federation_list**: URI dell'endpoint di listing della federazione (OpenID Federation `/list`). • **federation_fetch**: URI dell'endpoint fetch della federazione (OpenID Federation `/fetch`). • **federation_resolve**: URI dell'endpoint resolve della federazione (OpenID Federation `/resolve`). • **federation_trust_mark_status**: URI dell'endpoint di stato del Trust Mark. • **federation_historical_keys**: URI dell'endpoint degli JWK storici.
content_negotiation	OBBLIGATORIO. Array dei tipi di contenuto supportati dall'endpoint di discovery (es. `["application/json", "application/jwt"]`).

Struttura del payload JWT (in forma decodificata):

```
{
  "id": "urn:it-wallet-registry:it-wallet",
  "version": "1.0.0",
  "last_modified": "2024-03-15T10:30:00Z",
  "endpoints": {
    "claims_registry": "https://trust-anchor.eid-wallet.example.it/api/v1/claims",
    "authentic_sources": "https://trust-anchor.eid-wallet.example.it/api/v1/authentic-sources",
    "credential_catalog": "https://trust-anchor.eid-wallet.example.it/api/v1/.well-known/credential-catalog",
    "taxonomy": "https://trust-anchor.eid-wallet.example.it/api/v1/taxonomy",
    "schema_registry": "https://trust-anchor.eid-wallet.example.it/api/v1/schemas",
    "federation_list_endpoint": "https://trust-anchor.eid-wallet.example.it/list",
    "federation_fetch_endpoint": "https://trust-anchor.eid-wallet.example.it/federation_fetch_endpoint",
    "federation_resolve_endpoint": "https://trust-anchor.eid-wallet.example.it/resolve",
    "federation_trust_mark_status_endpoint": "https://trust-anchor.eid-wallet.example.it/trust_mark_status",
    "federation_trust_mark_list_endpoint": "https://dev.ta.wallet.ipzs.it/trust_mark_listing",
    "federation_trust_mark_endpoint": "https://dev.ta.wallet.ipzs.it/trust_mark",
    "federation_historical_keys_endpoint": "https://trust-anchor.eid-wallet.example.it/federation_historical_keys"
  },
  "content_negotiation": ["application/json", "application/jwt"]
}
```


7.3 Registro dei Claims

Il **Registro dei Claims** fornisce definizioni semantiche standardizzate per i singoli attributi delle Credenziali, i tipi di dati e le regole di validazione. Questo registro funge da fondamento semantico per la standardizzazione degli attributi delle Credenziali nell'intero ecosistema IT-Wallet, operando in coordinamento con il componente Tassonomia per la classificazione gerarchica.

L'Organismo di Vigilanza DEVE mantenere il Registro dei Claims per garantire la coerenza semantica e la conformità normativa nell'intero ecosistema. Il registro DEVE contenere:

- **Claims Standardizzati**: Definizioni semantiche per tutti gli attributi delle Credenziali con tipi di dati e regole di validazione.
- **Mappature di Interoperabilità**: Definizioni di alias per i claim che utilizzano terminologie diverse tra gli standard (es. `place_of_birth` di ISO18013-5 mappato al canonico `birth_place`).
- **Formati dei Dati**: Tipi di dati standardizzati (stringa, data, numerico, booleano, email, url, immagine, array, oggetto) con pattern di validazione.

Il Registro dei Claims DEVE garantire:

- **Coerenza Semantica**: Previene conflitti tra claim duplicati o sovrapposti nell'intero ecosistema.
- **Interoperabilità Transfrontaliera**: Garantisce la conformità EU e l'interpretazione coerente dei claim.
- **Validazione degli Schemi**: Fornisce definizioni autorevoli per la validazione dei claim in tutti gli scenari di Credenziale.
- **Allineamento Normativo**: Si coordina con il quadro normativo nazionale ed europeo.
- **Scenari Credential-Agnostic**: Supporta scenari in cui la **comodità dell'utente** e l'**efficienza operativa aziendale** sono prioritarie rispetto alla **conformità normativa** e alle **audit trail**.

Nota

Il Registro dei Claims definisce le proprietà semantiche dei singoli attributi, ma NON DEVE specificare le capacità di selective disclosure. La selective disclosure dipende dalle implementazioni del formato della Credenziale (SD-JWT, mDocs), dalle configurazioni tecniche dell'emittente e dal contesto di presentazione. Queste capacità sono specificate a livello di tipologia di Credenziale all'interno del Catalogo delle Credenziali Digitali e implementate durante i flussi di presentazione delle Credenziali.

7.3.1 Utilizzo del Registro dei Claims

Il Registro dei Claims DEVE supportare il ciclo di vita completo dell'ecosistema:

Durante il Processo di Onboarding:

- **Registrazione FA**: Le Fonti Autentiche dichiarano i claim disponibili dal registro standardizzato durante la registrazione delle capacità.
- **Registrazione EI**: Gli Emittenti di Credenziali selezionano le entità FA in base ai claim richiesti e registrano le tipologie di Credenziale per la pubblicazione nel catalogo.
- **Registrazione RP**: Le Relying Party specificano i requisiti di autorizzazione utilizzando domini/finalità per tipologie specifiche di Credenziale e/o attributi dell'Utente.

Durante le Attività Operative:

- **Emissione di Credenziali**: Le definizioni dei claim garantiscono una rappresentazione coerente dei dati tra le diverse tipologie di Credenziale.

- **Richieste di Presentazione**: Le RP fanno riferimento ai claim per la validazione degli schemi e la verifica dell'autorizzazione in scenari sia credential-specific che credential-agnostic.
- **Applicazione delle Politiche**: Le politiche di autorizzazione sfruttano le classificazioni per dominio/finalità per il controllo degli accessi.

7.3.2 Struttura del Registro dei Claims

Il Registro dei Claims mantiene definizioni tecniche neutrali rispetto alla lingua per la coerenza semantica nell'intero ecosistema. Le localizzazioni per gli utenti finali dei nomi e delle descrizioni dei claim sono fornite tramite bundle di localizzazione dedicati referenziati attraverso il campo `localization.base_uri`, abilitando un efficiente supporto multilingue senza compromettere l'integrità strutturale del registro.

Tabella 7.2: Campi di Primo Livello del Registro dei Claims

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del Registro dei Claims (es. `urn:claims:it-wallet`).
version	OBBLIGATORIO. La versione del Registro dei Claims (es. `1.0.0`).
last_modified	OBBLIGATORIO. Il timestamp che indica quando il Registro dei Claims è stato aggiornato l'ultima volta (es. `2026-03-06T00:00:00Z`).
localization	OBBLIGATORIO. Oggetto di configurazione della localizzazione contenente: • **default_locale**: Codice locale predefinito (es. `it`). • **available_locales**: Array dei codici locale supportati (es. `["en", "it"]`). • **base_uri**: URI base per il recupero del bundle di localizzazione (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/`). • **version**: Versione del formato del bundle di localizzazione.
claims	OBBLIGATORIO. Un Oggetto JSON in cui ogni chiave è il nome di un claim e ogni valore è un Oggetto JSON che descrive tale claim. Ogni oggetto claim contiene i parametri definiti nella tabella "Parametri di una Voce di Claim" di seguito.

Tabella 7.3: Parametri di una Voce di Claim

Nome Campo	Descrizione
description_l10n_id	OBBLIGATORIO. Chiave di localizzazione che fa riferimento alla descrizione leggibile del claim nel bundle di localizzazione (es. `claim.given_name.description`).
type	OBBLIGATORIO. Tipo di dato del claim. Valori supportati: `string`, `boolean`, `array`, `object`.
format	OPZIONALE. Qualificatore di formato semantico per i tipi stringa (es. `date` per le date ISO 8601, `uri`, `data` per dati binari Base64-encoded).
encoding	OPZIONALE. Codifica applicata al valore (es. `base64`). Presente quando `format` è `data`.
aliases	OPZIONALE. Array di nomi alternativi del claim utilizzati in altri standard che si mappano su questo claim canonico (es. `["birthdate"]` per `birth_date`, `["date_of_expiry"]` per `expiry_date`).
nested_claims	OPZIONALE. Array di nomi di claim che costituiscono le proprietà di un claim di tipo `object` (es. `["country", "locality", "region"]` per `place_of_birth`).
nested_item_claims	OPZIONALE. Array di nomi di claim che rappresentano le proprietà di ogni elemento in un claim di tipo `array` (es. `["vehicle_category_code", "issue_date", "expiry_date", "codes"]` per `driving_privileges`).
items	OPZIONALE. Oggetto JSON che descrive lo schema di ogni elemento in un claim di tipo `array` semplice (es. `{"type": "string"}` per `nationalities`).

Di seguito è fornito un esempio non normativo della struttura del Registro dei Claims:


```
{
    "id": "urn:claims:it-wallet",
    "version": "1.0.0",
    "last_modified": "2026-03-06T00:00:00Z",
    "localization": {
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/",
        "default_locale": "it",
        "version": "1.0.0"
    },
    "claims": {
        "academic_qualification_date": {
            "description_l10n_id": "claim.academic_qualification_date.description",
            "format": "date",
            "type": "string"
        },
        "academic_year": {
```

```
        "description_l10n_id": "claim.academic_year.description",
        "type": "string"
    },
    "address": {
        "description_l10n_id": "claim.address.description",
        "nested_claims": [
            "country",
            "house_number",
            "locality",
            "locality_fraction",
            "postal_code",
            "region",
            "street_address"
        ],
        "type": "object"
    },
    "birth_date": {
        "aliases": [
            "birthdate"
        ],
        "description_l10n_id": "claim.birth_date.description",
        "format": "date",
        "type": "string"
    },
    "birth_place": {
        "description_l10n_id": "claim.birth_place.description",
        "type": "string"
    },
    "codes": {
        "description_l10n_id": "claim.codes.description",
        "items": {
            "properties": {
                "code": {
                    "description_l10n_id": "claim.code.description",
                    "type": "string"
                },
                "sign": {
                    "description_l10n_id": "claim.sign.description",
                    "type": "string"
                },
                "value": {
                    "description_l10n_id": "claim.value.description",
                    "type": "string"
                }
            },
            "type": "object"
        },
```

```
        "nested_item_claims": [
            "code",
            "sign",
            "value"
        ],
        "type": "array"
    },
    "constant_attendance_allowance": {
        "description_l10n_id": "claim.constant_attendance_allowance.description",
        "type": "boolean"
    },
    "consulate_code": {
        "description_l10n_id": "claim.consulate_code.description",
        "type": "string"
    },
    "consulate_name": {
        "description_l10n_id": "claim.consulate_name.description",
        "type": "string"
    },
    "content": {
        "description_l10n_id": "claim.content.description",
        "encoding": "base64",
        "format": "data",
        "type": "string"
    },
    "country": {
        "description_l10n_id": "claim.country.description",
        "type": "string"
    },
    "cpia_school_year": {
        "description_l10n_id": "claim.cpia_school_year.description",
        "type": "string"
    },
    "degree_class": {
        "description_l10n_id": "claim.degree_class.description",
        "type": "string"
    },
    "degree_class_name": {
        "description_l10n_id": "claim.degree_class_name.description",
        "type": "string"
    },
    "degree_course_name": {
        "description_l10n_id": "claim.degree_course_name.description",
        "type": "string"
    },
    "degree_course_year": {
        "description_l10n_id": "claim.degree_course_year.description",
```

```
        "type": "string"
    },
    "document_iss_authority": {
        "description_l10n_id": "claim.document_iss_authority.description",
        "type": "string"
    },
    "document_iss_country": {
        "description_l10n_id": "claim.document_iss_country.description",
        "type": "string"
    },
    "document_number": {
        "description_l10n_id": "claim.document_number.description",
        "type": "string"
    },
    "driving_privileges": {
        "description_l10n_id": "claim.driving_privileges.description",
        "nested_item_claims": [
            "vehicle_category_code",
            "issue_date",
            "expiry_date",
            "codes"
        ],
        "type": "array"
    },
    "education_degrees": {
        "description_l10n_id": "claim.education_degrees.description",
        "nested_item_claims": [
            "academic_qualification_date",
            "degree_class",
            "degree_class_name",
            "degree_course_name",
            "institute_name",
            "programme_type_name",
            "qualification_grade_value",
            "qualification_name"
        ],
        "type": "array"
    },
    "education_enrollments": {
        "description_l10n_id": "claim.education_enrollments.description",
        "nested_item_claims": [
            "academic_year",
            "degree_class",
            "degree_class_name",
            "degree_course_name",
            "degree_course_year",
            "institute_name",
```

```
            "programme_type_name"
        ],
        "type": "array"
    },
    "educational_track": {
        "description_l10n_id": "claim.educational_track.description",
        "type": "string"
    },
    "effective_date": {
        "description_l10n_id": "claim.effective_date.description",
        "format": "date",
        "type": "string"
    },
    "expiry_date": {
        "aliases": [
            "date_of_expiry"
        ],
        "description_l10n_id": "claim.expiry_date.description",
        "format": "date",
        "type": "string"
    },
    "family_name": {
        "description_l10n_id": "claim.family_name.description",
        "type": "string"
    },
    "foreign_residence": {
        "description_l10n_id": "claim.foreign_residence.description",
        "type": "boolean"
    },
    "given_name": {
        "description_l10n_id": "claim.given_name.description",
        "type": "string"
    },
    "house_number": {
        "description_l10n_id": "claim.house_number.description",
        "type": "string"
    },
    "institute_code": {
        "description_l10n_id": "claim.institute_code.description",
        "type": "string"
    },
    "institute_name": {
        "description_l10n_id": "claim.institute_name.description",
        "type": "string"
    },
    "issue_date": {
        "description_l10n_id": "claim.issue_date.description",
```

```
        "format": "date",
        "type": "string"
    },
    "link_qr_code": {
        "description_l10n_id": "claim.link_qr_code.description",
        "format": "uri",
        "type": "string"
    },
    "locality": {
        "description_l10n_id": "claim.locality.description",
        "type": "string"
    },
    "locality_fraction": {
        "description_l10n_id": "claim.locality_fraction.description",
        "type": "string"
    },
    "nationalities": {
        "description_l10n_id": "claim.nationalities.description",
        "items": {
            "type": "string"
        },
        "type": "array"
    },
    "place_of_birth": {
        "description_l10n_id": "claim.place_of_birth.description",
        "nested_claims": [
            "country",
            "locality",
            "region"
        ],
        "type": "object"
    },
    "portrait": {
        "description_l10n_id": "claim.portrait.description",
        "encoding": "base64",
        "format": "data",
        "type": "string"
    },
    "postal_code": {
        "description_l10n_id": "claim.postal_code.description",
        "type": "string"
    },
    "professional_qualification": {
        "description_l10n_id": "claim.professional_qualification.description",
        "type": "string"
    },
    "programme_type_name": {
```

```
        "description_l10n_id": "claim.programme_type_name.description",
        "type": "string"
    },
    "qualification_grade_value": {
        "description_l10n_id": "claim.qualification_grade_value.description",
        "type": "string"
    },
    "qualification_name": {
        "description_l10n_id": "claim.qualification_name.description",
        "type": "string"
    },
    "qualification_type": {
        "description_l10n_id": "claim.qualification_type.description",
        "type": "string"
    },
    "qualification_type_description": {
        "description_l10n_id": "claim.qualification_type_description.description",
        "type": "string"
    },
    "qualification_year": {
        "description_l10n_id": "claim.qualification_year.description",
        "type": "string"
    },
    "region": {
        "description_l10n_id": "claim.region.description",
        "type": "string"
    },
    "restrictions_conditions": {
        "description_l10n_id": "claim.restrictions_conditions.description",
        "type": "string"
    },
    "school_attendance_type": {
        "description_l10n_id": "claim.school_attendance_type.description",
        "type": "string"
    },
    "school_code": {
        "description_l10n_id": "claim.school_code.description",
        "type": "string"
    },
    "school_course_year": {
        "description_l10n_id": "claim.school_course_year.description",
        "type": "string"
    },
    "school_name": {
        "description_l10n_id": "claim.school_name.description",
        "type": "string"
    },
```

```
        "school_qualifications": {
            "description_l10n_id": "claim.school_qualifications.description",
            "nested_item_claims": [
                "educational_track",
                "institute_code",
                "institute_name",
                "professional_qualification",
                "qualification_grade_value",
                "qualification_type",
                "qualification_type_description",
                "qualification_year",
                "school_code",
                "school_name",
                "school_year"
            ],
            "type": "array"
        },
        "school_year": {
            "description_l10n_id": "claim.school_year.description",
            "type": "string"
        },
        "street_address": {
            "description_l10n_id": "claim.street_address.description",
            "type": "string"
        },
        "tax_id_code": {
            "description_l10n_id": "claim.tax_id_code.description",
            "type": "string"
        },
        "un_distinguishing_sign": {
            "description_l10n_id": "claim.un_distinguishing_sign.description",
            "type": "string"
        },
        "vehicle_category_code": {
            "description_l10n_id": "claim.vehicle_category_code.description",
            "type": "string"
        }
    }
}
```

Nota

Per una gestione migliore e più efficiente della localizzazione delle informazioni contenute nel Registro dei Claims, un'Entità che lo consulta DOVREBBE:

- Scaricare la versione base del Registro dei Claims (compatta, senza localizzazioni)

utilizzando l'endpoint `.well-known/claims`.

- Determinare la lingua preferita dall'Utente.
- Scaricare solo i bundle di localizzazione necessari.
- Unire dinamicamente il contenuto localizzato con la struttura del Registro dei Claims.

Di seguito è fornito un esempio non normativo dell'output di un bundle di localizzazione:

```
{
  "claim.given_name.description": "Person's given name(s) as they appear on official documents.
",
  "claim.birth_date.description": "Date of birth, in ISO 8601 format (YYYY-MM-DD). Also known a
s birthdate.",
  "claim.driving_privileges.description": "Array of authorized vehicle categories with details.
",
  "...": "..."
}
```

I bundle di localizzazione DEVONO essere disponibili all'URI composto aggiungendo il codice locale e `.json` al valore `localization.base_uri` (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/claims/it.json`).

7.4 Registro delle Fonti Autentiche

L'Organismo di Vigilanza DEVE mantenere il Registro delle Fonti Autentiche per abilitare l'accesso coordinato ai dati e l'emissione di Credenziali nell'intero ecosistema. Il Registro FA DEVE contenere almeno:

- **Informazioni sull'Organizzazione**: Dettagli della persona giuridica, stato normativo e ruolo autorevole in domini specifici.
- **Capacità sui Dati**: Disponibilità dei claim dichiarati facendo riferimento alle definizioni standardizzate del Registro dei Claims con le corrispondenti classificazioni della Tassonomia.
- **Metodi di Integrazione**: Meccanismi tecnici di accesso (PDND).
- **Finalità Previste**: Tipologie di Credenziale supportate e contesti aziendali per il coordinamento FA-EI.
- **Garanzia della Qualità dei Dati**: Stato autorevole, frequenza di aggiornamento e capacità di audit trail.

Il Registro FA DEVE garantire:

- **Accesso Coordinato ai Dati**: Consente agli EI di scoprire i dati appropriati dalle Fonti Autentiche per l'emissione delle Credenziali.
- **Integrazione FA-EI**: Facilita i flussi di approvazione e il coordinamento dell'accesso ai dati tra le entità.
- **Garanzia della Qualità**: Mantiene lo stato autorevole e l'affidabilità dei dati nei diversi domini.
- **Conformità Normativa**: Supporta i requisiti di trasparenza della pubblica amministrazione e di coordinamento del settore privato.

Nota

Il Registro delle Fonti Autentiche è un registro pubblico che fornisce orientamento all'Emittente di Credenziali per il provisioning delle Credenziali.

7.4.1 Utilizzo del Registro delle Fonti Autentiche

Il Registro FA supporta il coordinamento dell'ecosistema durante l'intero ciclo di vita operativo:

Durante il Processo di Onboarding:

- **Auto-Dichiarazione FA**: Le Fonti Autentiche registrano le capacità prima che esistano tipologie di Credenziale nel catalogo.
- **Discovery EI**: Gli Emittenti di Credenziali ricercano le entità FA in base ai claim richiesti e alle tipologie di Credenziale previste.
- **Coordinamento delle Approvazioni**: Le entità FA valutano e approvano le richieste di accesso degli EI per la fornitura dei dati.

Durante le Attività Operative:

- **Risoluzione delle Sorgenti Dati**: I sistemi EI fanno riferimento al Registro FA per l'accesso in tempo reale ai dati durante l'emissione delle Credenziali.
- **Validazione della Qualità**: Le informazioni del Registro FA supportano la verifica dell'origine dei dati e i requisiti di audit.
- **Gestione dell'Integrazione**: Gli endpoint tecnici e i metodi di accesso consentono la comunicazione standardizzata FA-EI.

7.4.2 Coordinamento FA Pubblica vs FA Privata

L'architettura del Registro FA supporta diversi pattern di coordinamento che riflettono distinte esigenze operative:

1. **FA della Pubblica Amministrazione** (Integrazione Standardizzata): Gli enti pubblici forniscono dati autorevoli attraverso meccanismi regolamentati:

- **Integrazione PDND**: `"integration_method": "pdnd"` per l'accesso standardizzato ai dati pubblici.
- **Conformità Normativa**: Requisiti di piena trasparenza con pubblicazione pubblica nel catalogo.
- **Requisiti di Audit**: Tracciabilità completa per i processi di emissione di Credenziali pubbliche.

2. **FA del Settore Privato** (Integrazione Flessibile): Le entità private forniscono dati specializzati tramite accordi personalizzati:

- **API Personalizzate**: `"integration_method": "pdnd"` per l'accesso ai dati specifici del business.
- **Conformità Normativa**: Requisiti di piena trasparenza con pubblicazione pubblica nel catalogo.
- **Selective Disclosure**: Visibilità pubblica limitata con flussi di approvazione specifici per EI.
- **Flessibilità Aziendale**: Integrazione su misura a supporto di casi d'uso diversificati del settore privato.

Questo approccio consente sia la **trasparenza normativa** per la pubblica amministrazione sia la **flessibilità aziendale** per le entità del settore privato, mantenendo al contempo un accesso coordinato ai dati nell'intero ecosistema.

7.4.3 Struttura del Registro FA

Durante la registrazione, le Fonti Autentiche dichiarano le proprie capacità prima che le tipologie di Credenziale esistano nel catalogo. Questa dichiarazione costituisce la base per la successiva registrazione degli EI e la creazione delle tipologie di Credenziale.

Schema di Identificazione Univoca delle FA

A ciascuna Fonte Autentica DEVE essere assegnato un identificatore univoco che segue lo schema URL HTTPS definito di seguito. Questo identificatore è utilizzato per referenziare le entità FA nell'intero sistema di registro e nel Catalogo delle Credenziali Digitali, garantendo la coerenza con i pattern di identificazione delle entità di OpenID Federation.

Schema di Identificazione FA:

```
https://{organization_domain}[/{optional_path}]
```

Componenti dello Schema:

- **organization_domain**: Dominio DNS controllato dall'organizzazione
- **optional_path**: Componente di percorso aggiuntivo per servizi o dipartimenti specifici

L'identificatore FA DEVE seguire le seguenti regole normative:

1. **Protocollo HTTPS**: DEVE utilizzare lo schema HTTPS per sicurezza e verifica della fiducia
2. **Proprietà del Dominio**: L'organizzazione DEVE controllare il dominio DNS utilizzato nell'identificatore
3. **Unicità**: Garantita attraverso l'unicità del namespace DNS
4. **Stabilità**: DOVREBBE rimanere stabile nel tempo per evitare la rottura dei riferimenti
5. **Risolvibilità**: L'URL DOVREBBE essere risolvibile (ma non è obbligatorio che restituisca contenuto)

Esempi di identificatori FA conformi:

- `https://motorizzazione.gov.example`: Pubblico - Ministero dei Trasporti, Direzione Motorizzazione
- `https://registry.anpr.example`: Pubblico - Anagrafe Nazionale della Popolazione Residente
- `https://api.bank.example/auth-source`: Privato - Servizi Finanziari Banca Esempio

Parametri del Registro delle Fonti Autentiche

Il Registro delle Fonti Autentiche DEVE contenere i seguenti parametri per ciascuna Fonte Autentica registrata:

Tabella 7.4: Campi di Primo Livello del Registro delle Fonti Autentiche

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del Registro delle Fonti Autentiche (es. `urn:authentic-sources:it-wallet`).
version	OBBLIGATORIO. La versione del Registro delle Fonti Autentiche (es. `1.0.0`).
last_modified	OBBLIGATORIO. Il timestamp che indica quando il Registro delle Fonti Autentiche è stato aggiornato l'ultima volta (es. `2025-03-15T12:00:00Z`).
localization	OBBLIGATORIO. Oggetto di configurazione della localizzazione contenente: • **default_locale**: Codice locale predefinito (es. `it`). • **available_locales**: Array dei codici locale supportati (es. `["en", "it"]`). • **base_uri**: URI base per il recupero del bundle di localizzazione (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/`). • **version**: Versione del formato del bundle di localizzazione.
authentic_sources	OBBLIGATORIO. Un Array JSON in cui ogni voce è un Oggetto JSON che rappresenta un'entità Fonte Autentica. Ogni oggetto contiene i parametri definiti nella tabella "Parametri delle Fonti Autentiche" di seguito, inclusi identificazione dell'entità, informazioni organizzative, capacità sui dati e metodi di integrazione.

Tabella 7.5: Parametri delle Fonti Autentiche

Parametro	Tipo	Descrizione
entity_id	string	OBBLIGATORIO. Identificatore univoco che segue lo schema normativo: `https://{organization_domain}[/{optional_path}]`.
organization_info	oggetto JSON	OBBLIGATORIO. Dettagli della persona giuridica e metadati organizzativi.
organization_info.organization_name_l10n_id	string	OBBLIGATORIO. Chiave di localizzazione che fa riferimento al nome localizzato dell'organizzazione nel bundle di localizzazione (es. `authentic_source1.name`).
organization_info.organization_type	string	OBBLIGATORIO. Classificazione dell'entità: `"public"` o `"private"`.
organization_info.ipa_code	string	OBBLIGATORIO solo per FA Pubbliche. Codice di registrazione IPA per gli enti pubblici.

continues on next page

Tabella 7.5 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
organization_info.legal_identifier	string	OBBLIGATORIO. Identificativo di registrazione legale (Codice Fiscale/Partita IVA, o equivalente identificativo nazionale per entità straniere).
organization_info.homepage_uri	string	OBBLIGATORIO. URL della homepage dell'organizzazione.
organization_info.contacts	Array di stringhe	OBBLIGATORIO. Array di indirizzi email di contatto per almeno un referente di supporto utenti, uno applicativo e uno sistemistico.
organization_info.dpa_contact	string	OBBLIGATORIO. Indirizzo email del DPA della Fonte Autentica.
organization_info.policy_uri	string	OBBLIGATORIO. URL del documento di informativa sulla privacy.
organization_info.tos_uri	string	OPZIONALE. URL del documento dei termini di servizio.
organization_info.organization_country	string	OBBLIGATORIO. Codice paese ISO 3166-1 alpha-2 a due lettere dell'organizzazione.
organization_info.logo_uri	string	OPZIONALE. URL dell'immagine del logo dell'organizzazione.
organization_info.logo_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo per la verifica dell'integrità. OBBLIGATORIO se `logo_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
organization_info.logo_alt_text_l10n_id	string	OPZIONALE. Testo alternativo per l'immagine del logo dell'organizzazione.
organization_info.logo_extended_uri	string	OPZIONALE. URL dell'immagine del logo esteso dell'organizzazione.
organization_info.logo_extended_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo esteso per la verifica dell'integrità. OBBLIGATORIO se `logo_extended_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
organization_info.logo_extended_alt_text_l10n_id	string	OPZIONALE. Testo alternativo per l'immagine del logo esteso dell'organizzazione.
data_capabilities	Array di oggetti JSON	OBBLIGATORIO. Array contenente le specifiche delle capacità sui dati.
data_capabilities[].dataset_id	string	OBBLIGATORIO. Il *Dataset_id* nell'ambito della Fonte Autentica, che PUÒ essere utilizzato come parametro di query per il servizio `GetAttributeClaims`.
data_capabilities[].data_origin_l10n_id	string	OBBLIGATORIO. Chiave di localizzazione che fa riferimento al nome leggibile dell'origine o del dipartimento che fornisce i dati (es. `authentic_source1.dataset1.origin`).

continues on next page

Tabella 7.5 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
data_capabilities[]. intended_purposes	Array di stringhe	OBBLIGATORIO. Finalità aziendali soddisfatte, utilizzando gli identificatori di finalità della tassonomia (es. `["IDENTITY_VERIFICATION", "DRIVING_RIGHTS_VERIFICATION"]`).
data_capabilities[]. available_claims	Array di stringhe	OBBLIGATORIO. Claim disponibili da questa capacità dati.
data_capabilities[]. available_claims. claim_name	string	OBBLIGATORIO. Contiene il nome del claim.
data_capabilities[]. available_claims. order	number	OBBLIGATORIO. Definisce l'ordine in cui le informazioni vengono mostrate.
data_capabilities[]. available_claims. mandatory	boolean	OBBLIGATORIO. Definisce se un claim è sempre disponibile o meno.
data_capabilities[]. integration_method	string	OBBLIGATORIO. Framework di autorizzazione utilizzato per l'accesso ai dati. DEVE essere `"pdnd"`.
data_capabilities[]. integration_ endpoint	string	OPZIONALE. Punto di accesso al servizio (endpoint PDND).
data_capabilities[]. api_specification	string	OPZIONALE. URL del documento di specifica OAS3 per questa capacità dati.
data_capabilities[]. data_provision	oggetto JSON	OPZIONALE. Capacità di fornitura dei dati e specifiche temporali.
data_capabilities[]. data_provision. immediate_flow	boolean	OBBLIGATORIO. Indica se la Fonte Autentica supporta la fornitura immediata dei dati.
data_capabilities[]. data_provision. deferred_flow	boolean	OBBLIGATORIO. Indica se la Fonte Autentica supporta la fornitura differita dei dati.
data_capabilities[]. data_provision. max_response_ time_minutes	integer	CONDIZIONALE. Tempo massimo in minuti per la risposta della Fonte Autentica a una richiesta di fornitura dati differita. OBBLIGATORIO se `deferred_flow` è `true`.
data_capabilities[]. data_provision. notification_ methods	Array di stringhe	CONDIZIONALE. Array dei metodi di notifica supportati dalla Fonte Autentica per la fornitura dati differita, come `"push"`, `"poll"`. OBBLIGATORIO se `deferred_flow` è `true`.

continues on next page

Tabella 7.5 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
data_capabilities[].user_information_l10n_id	string	OPZIONALE. Chiave di localizzazione che fa riferimento a una stringa in formato Markdown con informazioni leggibili sulla capacità dati rilevanti per l'Utente (es. `authentic_source1.dataset1.userinfo`). Questa stringa DEVE essere fornita dalla Fonte Autentica al Trust Anchor durante l'onboarding. La formattazione Markdown può essere testo semplice o una combinazione di testo e link. Ad esempio, se il database della Fonte Autentica contiene solo dati registrati *dopo* una data specifica, questa informazione DEVE essere comunicata tramite questa chiave.
data_capabilities[].service_documentation_uri	string	OPZIONALE. URL che punta alla documentazione del servizio della Fonte Autentica.
data_capabilities[].update_frequency	string	OPZIONALE. Indica la frequenza con cui la Fonte Autentica aggiorna i propri dati. Valori possibili: `"real_time"` (aggiornamenti quasi in tempo reale, tipicamente entro minuti), `"daily"`, `"weekly"`, `"monthly"`, `"on_demand"`.
data_capabilities[].logo_uri	string	OPZIONALE. URL dell'immagine del logo relativa ai dati.
data_capabilities[].logo_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo per la verifica dell'integrità. OBBLIGATORIO se `logo_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
data_capabilities[].logo_alt_text_l10n_id	string	OPZIONALE. Testo alternativo per l'immagine del logo dell'organizzazione.
data_capabilities[].background_color	string	OPZIONALE. Valore stringa del colore di sfondo da visualizzare insieme ai dati.
data_capabilities[].contacts	Array di Oggetti	OPZIONALE. Array di indirizzi email di contatto del servizio clienti o dei canali di supporto per gli utenti riferiti allo specifico dataset. Ogni oggetto contiene: • **type**: OBBLIGATORIO. Indica il tipo di informazione di contatto e DEVE essere impostato a `email`, `telephone` oppure `url`. • **value**: OBBLIGATORIO. Contiene il valore dell'informazione di contatto corrispondente al tipo specificato. • **description**: OPZIONALE. Contiene una descrizione testuale associata all'informazione di contatto.

Nota

Per ulteriori dettagli sulle funzionalità richieste e sul risultato atteso in termini di esperienza utente, si rimanda alla Sezione *Ottenimento degli Attestati Elettronici di Attributi* (pagina 46) per il parametro `data_capabilities.user_information` e alla Sezione *Focus sugli Attestati Elettronici di Attributi* (pagina 61) per i parametri `organization_info.logo_uri`, `organization_info.logo_extended_uri`, `data_capabilities.logo_uri`, `data_capabilities.background_color` e `data_capabilities.available_claims.order`.

Esempio di Registro FA

Di seguito è fornito un esempio non normativo della struttura del Registro FA:


```
{
    "id": "urn:authentic-sources:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "localization": {
        "default_locale": "it",
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/",
        "version": "1.0.0"
    },
    "authentic_sources": [
        {
            "entity_id": "https://motorizzazione.gov.example",
            "organization_info": {
                "contacts": [
                    "registry@motorizzazione.gov.example",
                    "technical-support@motorizzazione.gov.example"
                ],
                "dpa_contact": "dpa@motorizzazione.gov.example",
                "homepage_uri": "https://www.gov.example/transport",
                "ipa_code": "m_inf",
                "legal_identifier": "80192770587",
                "logo_extended_uri": "https://trust-registry.eid-wallet.example.it/logos/m_inf/extended-image.svg",
                "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3",
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/m_inf/image.svg",
```

```
            "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e
998e86f7f7a27ae3",
            "organization_country": "IT",
            "organization_name": "Ministero dei Trasporti - Direzione generale per la Motor
izzazione",
            "organization_name_l10n_id": "authentic_source1.name",
            "organization_type": "public",
            "policy_uri": "https://www.gov.example/transport/privacy-policy",
            "tos_uri": "https://www.gov.example/transport/terms-of-service"
        },
        "data_capabilities": [
            {
                "api_specification": "https://docs.gov.example/transport/api-oas3.yaml",
                "available_claims": [
                    {
                        "claim_name": "given_name",
                        "mandatory": true,
                        "order": 1
                    },
                    {
                        "claim_name": "family_name",
                        "mandatory": true,
                        "order": 2
                    },
                    {
                        "claim_name": "birth_date",
                        "mandatory": true,
                        "order": 3
                    },
                    {
                        "claim_name": "birth_place",
                        "mandatory": true,
                        "order": 4
                    },
                    {
                        "claim_name": "issue_date",
                        "mandatory": true,
                        "order": 5
                    },
                    {
                        "claim_name": "document_iss_country",
                        "mandatory": true,
                        "order": 6
                    },
                    {
                        "claim_name": "document_iss_authority",
                        "mandatory": true,
```

```
            "order": 7
        },
        {
            "claim_name": "expiry_date",
            "mandatory": true,
            "order": 8
        },
        {
            "claim_name": "document_number",
            "mandatory": true,
            "order": 9
        },
        {
            "claim_name": "portrait",
            "mandatory": true,
            "order": 10
        },
        {
            "claim_name": "un_distinguishing_sign",
            "mandatory": true,
            "order": 11
        },
        {
            "claim_name": "restrictions_conditions",
            "mandatory": false,
            "order": 12
        },
        {
            "claim_name": "driving_privileges",
            "mandatory": true,
            "order": 13
        }
    ],
    "background_color": "#12107c",
    "contacts": [
        {
            "type": "email",
            "value": "citizen-support@motorizzazione.gov.example"
        }
    ],
    "data_origin_l10n_id": "authentic_source1.dataset1.origin",
    "data_provision": {
        "deferred_flow": false,
        "immediate_flow": true
    },
    "dataset_id": "38832801",
```

```
                    "integration_endpoint": "https://api.gov.example/transport/driving-license"
,
                    "integration_method": "pdnd",
                    "intended_purposes": [
                        "IDENTITY_VERIFICATION",
                        "PERSON_IDENTIFICATION",
                        "DRIVING_RIGHTS_VERIFICATION",
                        "LAW_ENFORCEMENT_CONTROLS"
                    ],
                    "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38832801/im
age.svg",
                    "logo_uri#integrity": "sha-256-b1946ac92492d2347c6235b4d2611184e2a3f8b8c8f8
f8e8f8f8f8f8f8f8f8f8",
                    "service_documentation_uri": "https://docs.gov.example/transport/api-docs",
                    "update_frequency": "real_time",
                    "user_information_l10n_id": "authentic_source1.dataset1.userinfo"
                }
            ]
        },
        {
            "entity_id": "https://api.bank.example/auth-source",
            "organization_info": {
                "contacts": [
                    "digital-credentials@bank.example",
                    "api-support@bank.example"
                ],
                "dpa_contact": "dpa@bank.example",
                "homepage_uri": "https://www.bank.example",
                "legal_identifier": "12345678901",
                "logo_extended_uri": "https://trust-registry.eid-wallet.example.it/logos/123456
78901/extended-image.svg",
                "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3
fff1fa07e998e86f7f7a27ae3",
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/12345678901/ima
ge.svg",
                "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e
998e86f7f7a27ae3",
                "organization_country": "IT",
                "organization_name": "Banca Esempio SpA",
                "organization_name_l10n_id": "authentic_source2.name",
                "organization_type": "private",
                "policy_uri": "https://www.bank.example/privacy-policy",
                "tos_uri": "https://www.bank.example/terms-of-service"
            },
            "data_capabilities": [
                {
                    "api_specification": "https://api.bank.example/docs/psd2-openapi.yaml",
```

```
"available_claims": [
    {
        "claim_name": "given_name",
        "mandatory": true,
        "order": 1
    },
    {
        "claim_name": "family_name",
        "mandatory": true,
        "order": 2
    },
    {
        "claim_name": "birth_date",
        "mandatory": false,
        "order": 3
    },
    {
        "claim_name": "tax_id_code",
        "mandatory": true,
        "order": 4
    },
    {
        "claim_name": "account_holder_name",
        "mandatory": true,
        "order": 5
    },
    {
        "claim_name": "iban",
        "mandatory": true,
        "order": 6
    },
    {
        "claim_name": "account_status",
        "mandatory": false,
        "order": 7
    },
    {
        "claim_name": "account_opening_date",
        "mandatory": false,
        "order": 8
    },
    {
        "claim_name": "bank_name",
        "mandatory": true,
        "order": 9
    },
    {
```

```
                        "claim_name": "bank_code",
                        "mandatory": true,
                        "order": 10
                    }
                ],
                "background_color": "#003366",
                "contacts": [
                    {
                        "type": "email",
                        "value": "digital-credentials@bank.example"
                    }
                ],
                "data_origin_l10n_id": "authentic_source2.dataset1.origin",
                "data_provision": {
                    "deferred_flow": false,
                    "immediate_flow": true
                },
                "dataset_id": "38854801",
                "integration_endpoint": "https://api.bank.example/psd2/v1/accounts",
                "integration_method": "pdnd",
                "intended_purposes": [
                    "PAYMENT_AUTHORIZATION",
                    "ELIGIBILITY_VERIFICATION"
                ],
                "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38854801/im
age.svg",
                "logo_uri#integrity": "sha-256-d8e8fca2dc0f896fd7cb4cb0031ba249e3f8b8c8f8f8
e8f8f8f8f8f8f8f8f8f9",
                "service_documentation_uri": "https://docs.bank.example/psd2-api",
                "update_frequency": "daily",
                "user_information_l10n_id": "authentic_source2.dataset1.userinfo"
            },
            {
                "api_specification": "https://api.bank.example/docs/psd2-openapi.yaml",
                "available_claims": [
                    {
                        "claim_name": "given_name",
                        "mandatory": true,
                        "order": 1
                    },
                    {
                        "claim_name": "family_name",
                        "mandatory": true,
                        "order": 2
                    },
                    {
                        "claim_name": "tax_id_code",
```

```
                "mandatory": true,
                "order": 3
            },
            {
                "claim_name": "transaction_history",
                "mandatory": true,
                "order": 4
            },
            {
                "claim_name": "average_balance",
                "mandatory": false,
                "order": 5
            },
            {
                "claim_name": "credit_score_indicator",
                "mandatory": false,
                "order": 6
            }
        ],
        "background_color": "#003366",
        "contacts": [
            {
                "type": "email",
                "value": "digital-credentials@bank.example"
            }
        ],
        "data_origin_l10n_id": "authentic_source2.dataset2.origin",
        "data_provision": {
            "deferred_flow": true,
            "immediate_flow": false,
            "max_response_time_minutes": 1440,
            "notification_methods": [
                "push",
                "poll"
            ]
        },
        "dataset_id": "38854802",
        "integration_endpoint": "https://api.bank.example/psd2/v1/transactions",
        "integration_method": "pdnd",
        "intended_purposes": [
            "PROOF_OF_PAYMENT",
            "RECURRING_PAYMENT_MANAGEMENT"
        ],
        "logo_uri": "https://trust-registry.eid-wallet.example.it/logos/38854802/im
age.svg",
        "logo_uri#integrity": "sha-256-f4a2b7e1dc0f896fd7cb4cb0031ba249e3f8b8c8f8f8
e8f8f8f8f8f8f8f8f8fb",
```

```
                "service_documentation_uri": "https://docs.bank.example/psd2-api",
                "update_frequency": "weekly",
                "user_information_l10n_id": "authentic_source2.dataset2.userinfo"
            }
        ]
    }
  ]
}
```

Nota

Per una gestione migliore e più efficiente della localizzazione delle informazioni contenute nel Registro delle Fonti Autentiche, un'Entità che lo consulta DOVREBBE:

- Scaricare la versione base del Registro delle Fonti Autentiche (compatta, senza localizzazioni) utilizzando l'endpoint `.well-known/authentic-sources`.
- Determinare la lingua preferita dall'Utente.
- Scaricare solo i bundle di localizzazione necessari.
- Unire dinamicamente il contenuto localizzato con la struttura del Registro delle Fonti Autentiche.

Di seguito è fornito un esempio non normativo dell'output di un bundle di localizzazione:

```
{
  "authentic_source1.name": "Ministero delle infrastrutture e dei trasporti",
  "authentic_source1.dataset1.origin": "MIT -- Direzione Generale per la Motorizzazione",
  "authentic_source1.dataset1.userinfo": "###### Patente di Guida\nSono disponibili le patenti rilasciate dopo il 1° gennaio 2020. Per le patenti più vecchie, contattare l'ufficio motorizzazione locale.",
  "authentic_source2.name": "Banca Esempio SpA",
  "authentic_source2.dataset1.origin": "Esempio origine dei dati 1",
  "authentic_source2.dataset1.userinfo": "###### Informazioni sulla disponibilità dei dati\nL'accesso ai dati finanziari richiede il consenso del cliente ed è soggetto alla normativa PSD2. Le informazioni sui conti sono disponibili solo per i conti attivi.",
  "...": "..."
}
```

I bundle di localizzazione DEVONO essere disponibili all'URI composto aggiungendo il codice locale e `.json` al valore `localization.base_uri` definito nel registro. Ciascun bundle locale DEVE essere accessibile seguendo il pattern di denominazione **{locale_code}.json**, dove **{locale_code}** è sostituito con il codice locale corrispondente dall'array **available_locales**.

Un esempio non normativo dell'URI di localizzazione italiana per il bundle sarebbe **https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/it.json**.

7.4.4 Coordinamento FA-EI

A seguito della registrazione FA, il Registro FA consente agli Emittenti di Credenziali di individuare entità FA idonee e richiedere l'approvazione dell'integrazione. Questo processo di coordinamento è descritto in dettaglio in *Processo di Autorizzazione dalla Fonte Autentica all'Emittente di Credenziali* (pagina 255).

7.5 Registro della Federazione

Il **Registro della Federazione** fornisce l'infrastruttura di fiducia crittografica per tutti i partecipanti all'ecosistema IT-Wallet. Il Registro della Federazione mantiene l'elenco autorevole delle entità fidate e il loro stato operativo utilizzando endpoint specifici della federazione come definito in *Endpoint API di Federazione* (pagina 123).

7.5.1 Ruolo di Integrazione del Registro

All'interno dell'architettura del Registro di Sistema IT-Wallet, il Registro della Federazione funge da **livello di validazione della fiducia** per:

1. **Autenticazione delle Entità**: Valida l'identità crittografica di tutti i partecipanti prima delle operazioni di registro
2. **Verifica della Catena di Fiducia**: Fornisce il fondamento crittografico per la validazione delle entità Emittenti di Credenziali, Relying Party e Fornitori di Wallet
3. **Verifica della Conformità**: Mantiene i Trust Mark che attestano la conformità normativa e lo stato operativo

7.5.2 Accesso al Registro della Federazione

Le operazioni del Registro della Federazione sono accessibili attraverso gli endpoint di federazione del Trust Anchor come dettagliato in *Endpoint API di Federazione* (pagina 123). L'architettura di discovery del registro fornisce informazioni sugli endpoint di federazione tramite l'endpoint di discovery del registro descritto in *Endpoint di Discovery del Registro* (pagina 160).

Nota

Gli endpoint di federazione sono disponibili sia attraverso il meccanismo di discovery del registro (per l'accesso unificato al registro) sia attraverso la Configurazione dell'Entità del Trust Anchor all'indirizzo `.well-known/openid-federation` (per operazioni specifiche della federazione). Entrambe le sorgenti forniscono gli stessi URL degli endpoint ma servono diversi

pattern di discovery: la discovery del registro per l'orientamento iniziale nell'ecosistema, la Configurazione dell'Entità per la conformità standard a OpenID Federation 1.0.

Per le specifiche tecniche complete dei protocolli di federazione, le configurazioni delle entità, i meccanismi di valutazione della fiducia e la validazione della catena di fiducia, vedere *L'Infrastruttura di Trust* (pagina 117).

7.6 Catalogo degli Attestati Elettronici

Il Catalogo delle Credenziali Digitali è il registro di tutte le Credenziali Digitali disponibili riconosciute nell'ecosistema IT-Wallet. È pubblicato dal Trust Anchor ed è pubblicamente disponibile per tutte le Entità tramite un endpoint specializzato della Federazione. Funge da unico punto di riferimento per tutti gli attori coinvolti nel processo di emissione, verifica e utilizzo delle Credenziali Digitali.

Il Catalogo delle Credenziali Digitali si prefigge di:

1. Facilitare la discovery delle Credenziali Digitali per gli Utenti.
2. Standardizzare la descrizione tecnica e funzionale delle Credenziali Digitali.
3. Abilitare l'interoperabilità tra diversi Emittenti e Relying Party.
4. Semplificare il processo di integrazione per i Fornitori di Wallet e le Relying Party.
5. Garantire la fiducia nell'ecosistema attraverso informazioni verificabili e attendibili.
6. Fornire trasparenza sull'ecosistema delle Credenziali Digitali disponibili.

Le principali Entità coinvolte nel Catalogo delle Credenziali Digitali sono:

- **Trust Anchor**: Gestisce e mantiene il Catalogo delle Credenziali Digitali, garantendone l'autenticità e l'integrità.
- **Organismo di Vigilanza**: Interagisce con il Trust Anchor e il Catalogo delle Credenziali Digitali per monitorare la fase di registrazione garantendo sicurezza e privacy secondo la normativa nazionale/europea, mantenendo tutte le informazioni affidabili e aggiornate.
- **Emittenti di Credenziali Digitali**: Le entità autorizzate a emettere Credenziali Digitali, registrandole nel Catalogo.
- **Relying Party**: Utilizzano il Catalogo delle Credenziali Digitali per raccogliere tutte le informazioni necessarie sulle Credenziali Digitali che intendono richiedere durante la fase di presentazione.
- **Fornitori di Wallet**: Accedono al Catalogo delle Credenziali Digitali per identificare le Credenziali Digitali disponibili e recuperare tutte le informazioni necessarie per integrarle nelle proprie Soluzioni Wallet.

- **Utenti**: Gli Utenti che utilizzano indirettamente il Catalogo delle Credenziali Digitali attraverso le proprie Istanze Wallet per scoprire e richiedere Credenziali Digitali.
- **Fonti Autentiche**: Le Entità che detengono i dati originali attestati nelle Credenziali Digitali. Forniscono supporto agli Emittenti nella registrazione delle Credenziali Digitali nel Catalogo.

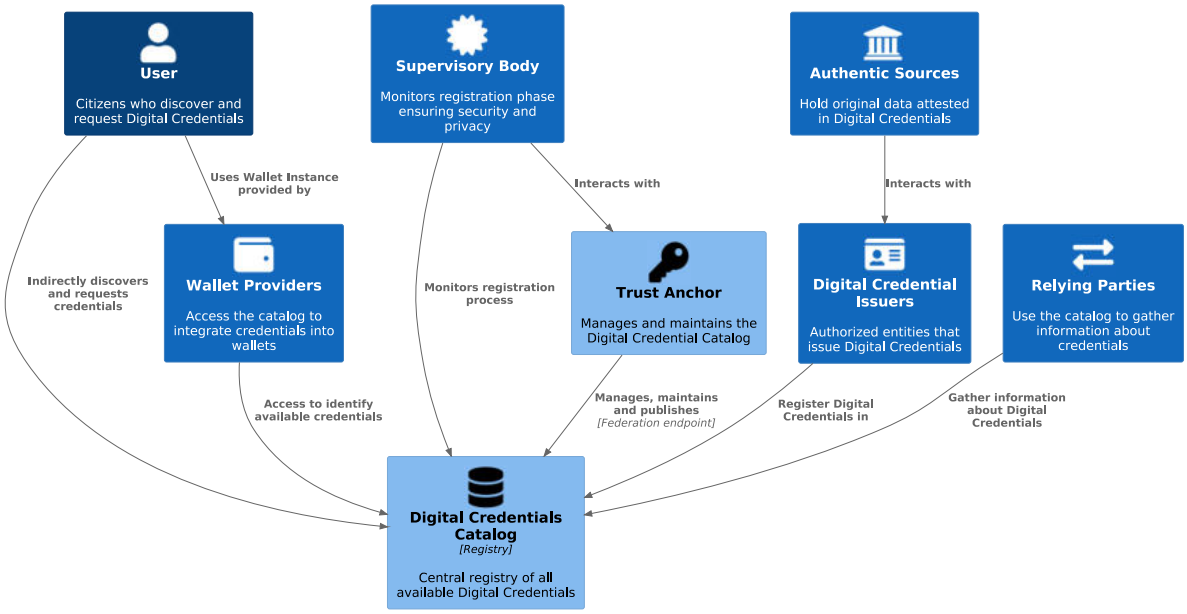


Fig. 7.1: Diagramma Entità-Relazione del Catalogo delle Credenziali Digitali.

La seguente tabella riepiloga le principali informazioni che DEVONO essere fornite dal Catalogo delle Credenziali Digitali:

Tabella 7.6: Catalogo delle Credenziali Digitali - Informazioni principali

Informazioni relative a	Descrizione
Metadati della Credenziale Digitale	Informazioni identificative essenziali e caratteristiche della Credenziale Digitale, tra cui: • **Identificatore univoco della Credenziale**: Una stringa identificativa univoca per ciascuna Credenziale Digitale. • **Metodi di autenticazione dell'Utente**: Meccanismi di autenticazione dell'Utente utilizzati per richiedere la Credenziale Digitale, se richiesti dagli Emittenti o dalle Fonti Autentiche. • **Livello Minimo di Garanzia**: Il Livello Minimo di Garanzia richiesto per l'affidabilità della Credenziale Digitale. DEVE tener conto del Livello di Garanzia dell'autenticazione dell'Utente, ove applicabile, e dell'Istanza Wallet.
Emittenti della Credenziale Digitale	Dettagli sull'organizzazione autorizzata a emettere la Credenziale Digitale, quali: • **Identificatori dell'Emittente**: Identificatore univoco dell'emittente della Credenziale Digitale. • **Tipologia dell'Emittente**: Classificazione come fornitore PID, (Q)EAA o Pub-EAA. • **Informazioni aggiuntive**: Dettagli organizzativi inclusi nome, codice e informazioni di contatto.
Fonti Autentiche Specifica Tecnica	Informazioni sulla fonte di dati autorevole. Dettagli tecnici, tra cui: • **Schemi della Credenziale Digitale**: Specifiche del framework e della struttura. • **Formati della Credenziale Digitale**: Standard di formato e codifica dei dati. • **Politica di autenticazione**: Metodi e requisiti per la verifica.

continues on next page

Tabella 7.6 – continua dalla pagina precedente

Informazioni relative a	Descrizione
Termini di Utilizzo	Condizioni e limitazioni per l'utilizzo della Credenziale Digitale, quali: • **Validità della Credenziale**: Periodo di tempo durante il quale la Credenziale Digitale è valida e, ove applicabile, meccanismi e dettagli tecnici per invalidare le Credenziali Digitali (metodi di revoca/sospensione). • **Politica di restrizione**: Se applicabile, regole che disciplinano l'utilizzo e le limitazioni della Credenziale Digitale secondo la normativa nazionale. Viene utilizzata, ad esempio, per specificare se solo specifiche tipologie di entità legali, come i Fornitori Pub-EAA e le Soluzioni Wallet pubbliche, sono autorizzate a emettere e ottenere la Credenziale Digitale. • **Politica di prezzo**: Informazioni relative ai modelli di prezzo della Credenziale Digitale, come *free*, *issuance_based*, *verification_based*. • **Finalità della Credenziale Digitale**: Informazioni relative alle finalità consentite per cui la Credenziale Digitale può essere utilizzata. Ogni tipologia di Credenziale Digitale può essere utilizzata per più finalità.

Il Trust Anchor DEVE pubblicare e mantenere aggiornate tutte le informazioni all'endpoint *.well-known* del Catalogo delle Credenziali Digitali garantendo affidabilità, autenticità e integrità dei dati. In particolare, il Catalogo delle Credenziali Digitali DEVE essere disponibile attraverso l'endpoint `.well-known/credential-catalog`. DEVE supportare `application/jose` come content-type.

Di seguito è riportato un esempio non normativo.

```
GET /.well-known/credential-catalog HTTP/1.1
Host: trust-anchor.eid-wallet.example.it
Accept: application/jose

HTTP/1.1 200 OK
Content-Type: application/jose

eyJhbGciOiJSUzI1NiIsImtpZCI6ImV4YW1w...
```

Nella sezione *Struttura del Catalogo degli Attestati Elettronici* (pagina 206) è fornito un esempio del Catalogo delle Credenziali Digitali decodificato in JSON.

7.6.1 Gerarchia delle Credenziali Digitali

Le Credenziali Digitali riconosciute nell'ecosistema IT-Wallet sono classificate e standardizzate secondo il seguente modello gerarchico multilivello, progettato per migliorare la chiarezza semantica, la discovery delle credenziali e la compatibilità con i flussi di verifica sia credential-specific che claim-based.

La gerarchia è definita come segue:

Dominio

Un **Dominio** rappresenta un'area tematica di alto livello che raggruppa famiglie di Credenziali afferenti allo stesso contesto generale (es. Identità, Salute, Istruzione, Mobilità). I Domini forniscono un livello organizzativo di primo piano.

Classe di Credenziale

Una **Classe di Credenziale** rappresenta una famiglia di Credenziali che condividono natura, funzione o struttura simili (es. Documenti di Identità, Certificati di Stato Civile).

Ogni Classe DOVREBBE definire:

- un identificatore di Classe stabile (URI),
- la semantica attesa della Famiglia di Credenziali.

Le Classi consentono alle Relying Party e alle Soluzioni Wallet di richiedere o abbinare le Credenziali in base alla loro categoria tipologica.

Tipologia di Credenziale

Una **Tipologia di Credenziale** rappresenta una Credenziale specifica all'interno di una Classe (es. Credenziale di Viaggio Digitale, Certificato di Nascita, Patente di Guida Mobile). Ogni Tipologia di Credenziale DEVE includere:

- un identificatore univoco,
- l'identificatore dell'Emittente della Credenziale,
- l'insieme degli Attributi che possono essere inclusi nelle presentazioni.

Le Tipologie di Credenziale consentono il targeting preciso per i flussi di verifica conformi o normativamente mandatati.

Finalità (Intento di Verifica)

Una **Finalità (Intento di Verifica)** descrive *perché* una credenziale può essere richiesta da una Relying Party (es. Verifica dell'Identità, Verifica dell'Età, Idoneità a servizi specifici). Le Finalità DEVONO descrivere **gli esiti della verifica**. Ogni Tipologia di Credenziale DEVE dichiarare il proprio Dominio, Classe e Finalità supportate.

Le tabelle seguenti forniscono esempi non esaustivi che illustrano le relazioni tra Domini, Classi di Credenziale e Tipologie di Credenziale, seguiti dalla loro mappatura sulle Finalità di verifica. Domini, Classi, Credenziali specifiche e Finalità di verifica aggiuntive **POSSONO** essere aggiunti nel tempo con l'evolversi dell'ecosistema IT-Wallet.

Tabella 7.7: Tassonomia delle Credenziali Digitali: Gerarchia e Classificazione

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
IDENTITÀ	Credenziali che stabiliscono o confermano l'identità legale di una persona e il suo status personale, civile o giuridico.	• Documenti di Identità • Certificati Anagrafici e di Stato Civile • Stato Economico e Giuridico	• Credenziale di Viaggio Digitale • Patente di Guida Mobile (solo Italia) • Codice Fiscale / Tessera Sanitaria • Certificazione dell'Età • Certificato di Nascita • Certificato di Residenza • Certificato di Stato di Famiglia • Certificato di Matrimonio • Certificato di Cittadinanza • ISEE (Indicatore della Situazione Economica Equivalente) • Permesso di Soggiorno • Certificato dei Carichi Pendenti • Certificato del Casellario Giudiziale

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
CASA E FAMIGLIA	Credenziali che attestano la composizione del nucleo familiare, la residenza e i rapporti giuridici o fiscali legati all'abitazione.	• Documenti Catastali e Immobiliari • Documenti Familiari • Documenti Tributari Locali	• Atto di Compravendita • Visura Catastale • Planimetria Catastale • Certificato Catastale • Codice Fiscale/Tessera Sanitaria dei Figli • Certificato di Nascita • Certificato di Stato di Famiglia • IMU (Imposta Municipale Propria) • TARI (Tassa sui Rifiuti)
ISTRUZIONE	Credenziali che attestano risultati scolastici, titoli accademici e formazione professionale.	• Titoli di Studio • Certificazioni Professionali	• Diploma di Licenza Media • Diploma di Istruzione Secondaria Superiore • Laurea Triennale • Laurea Magistrale • Master Universitario • Dottorato di Ricerca • Albi Professionali (es. architetto, avvocato) • Attestati di Formazione Professionale • Certificazioni Linguistiche (es. IELTS) • Qualifiche Accademiche (es. Europass)

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
SALUTE	Credenziali relative alla copertura sanitaria, allo stato di salute e a certificazioni di carattere sanitario.	• Certificazioni e Idoneità • Cartelle Cliniche	• Tessera Sanitaria (TEAM) • Tessera Europea di Assicurazione Malattia (CED) • Certificato di Disabilità • Certificato di Vaccinazione • Certificato di Idoneità Sportiva • Certificato di Idoneità al Lavoro • Prescrizioni Mediche • Referto Medico Digitale

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
FINANZIARIO	Credenziali relative a strumenti di pagamento, autorizzazioni finanziarie e prove di pagamento.	• Strumenti di Pagamento • Credenziali e Autorizzazioni di Pagamento • Pagamenti e Tributi Pubblici • Pagamenti Ricorrenti e Abbonamenti	• Carta di Pagamento Digitale (debito / credito / prepagata) • Carta Virtuale • Conto Bancario (IBAN) • Credenziale di Strong Customer Authentication (SCA) • Ricevuta di Pagamento • Bollo Digitale • Certificato di Pagamento di Tasse e Tributi • Mandato di Abbonamento • Credenziale di Pagamento Ricorrente
CULTURA E TEMPO LIBERO	Credenziali che attestano l'appartenenza, l'affiliazione o la partecipazione a programmi culturali o ricreativi.	• Carte e Benefici Culturali • Programmi di Fidelizzazione e Associativi	• Carta della Cultura • Abbonamenti Annuali ai Musei • Carta Cinema • Carta Museo • Tessere di Associazione • Tessera Bibliotecaria • City Pass

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
LAVORO	Credenziali che attestano rapporti di lavoro, status professionale e contribuzione previdenziale.	• Documenti di Lavoro • Stato Occupazionale • Accesso	• Contratto di Lavoro Digitale • Curriculum Vitae (CV) • Permesso di Soggiorno • Certificato di Stato Occupazionale • Estratto Contributivo INPS • Badge di Accesso Fisico

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
MOBILITÀ E VIAGGI	Credenziali che attestano diritti di mobilità, stato del veicolo e titolarità in ambito di viaggio.	• Patenti e Autorizzazioni • Documenti del Veicolo • Abbonamenti al Trasporto • Documenti di Viaggio • Assicurazione Viaggio • Prenotazioni • Sconti e Agevolazioni	• Patente di Guida Mobile • Patente Nautica • Certificato di Proprietà del Veicolo • Assicurazione RCA Digitale • Certificato di Revisione del Veicolo • Carta Verde / Assicurazione Internazionale • Abbonamento ai Trasporti Pubblici • Abbonamento Pedaggi Stradali • Credenziale di Viaggio Digitale • Biglietti di Viaggio (aereo, treno, ecc.) • Polizza di Assicurazione Viaggio • Prenotazione Hotel • Carte Sconto • Agevolazioni Turistiche

continues on next page

Tabella 7.7 – continua dalla pagina precedente

Dominio	Descrizione	Classe di Credenziale	Tipologia di Credenziale
BONUS	Credenziali che attestano il diritto a benefici economici, incentivi o voucher.	• Benefici Economici e Sussidi • Incentivi e Voucher • Bonus Salute e Benessere	• Credenziale Assegno Familiare • Credenziale Indennità di Disoccupazione • Voucher Digitale • Credenziale Incentivo all'Acquisto • Credenziale Idoneità Cashback • Credenziale Bonus Sanitario • Voucher Supporto Salute Mentale • Bonus Sport e Attività Fisica

Tabella 7.8: Tabella 2: Mappatura tra Classi di Credenziale e Finalità

Classe di Credenziale	Finalità Supportate
Documenti di Identità	• Verifica dell'identità • Verifica dell'età • Identificazione della persona
Certificati Anagrafici e di Stato Civile	• Verifica dello stato civile • Diritto di residenza • Verifica della composizione del nucleo familiare

continues on next page

Tabella 7.8 – continua dalla pagina precedente

Classe di Credenziale	Finalità Supportate
Stato Economico e Giuridico	• Idoneità a servizi o benefici • Verifica dello stato giuridico • Verifica del casellario giudiziale
Documenti Catastali e Immobiliari	• Verifica di residenza e nucleo familiare • Verifica della proprietà immobiliare • Conformità urbanistica e catastale
Documenti Familiari	• Verifica della composizione del nucleo familiare • Idoneità ai servizi sociali basati sul nucleo familiare
Documenti Tributari Locali	• Conformità agli obblighi tributari locali • Verifica dello stato dell'imposta sulla proprietà
Titoli di Studio	• Verifica del titolo e del grado accademico • Idoneità ai percorsi formativi
Certificazioni Professionali	• Verifica dell'abilitazione professionale • Valutazione delle competenze ai fini lavorativi
Certificazioni e Idoneità	• Verifica dello stato vaccinale • Verifica dello stato di idoneità • Accesso ad aree con restrizioni sanitarie

continues on next page

Tabella 7.8 – continua dalla pagina precedente

Classe di Credenziale	Finalità Supportate
Cartelle Cliniche	• Accesso ai servizi sanitari • Condivisione della documentazione medica • Validazione della storia clinica
Strumenti di Pagamento	• Autorizzazione al pagamento • Esecuzione del pagamento • Prova di pagamento
Credenziali e Autorizzazioni di Pagamento	• Gestione delle autorizzazioni finanziarie • Strong Customer Authentication (SCA)
Pagamenti e Tributi Pubblici	• Prova di pagamento delle tasse • Prova di pagamento dei tributi • Validazione del bollo digitale
Pagamenti Ricorrenti e Abbonamenti	• Gestione dei pagamenti ricorrenti • Verifica del mandato di abbonamento
Carte e Benefici Culturali	• Accesso ai servizi culturali • Accesso ai servizi ricreativi • Applicazione degli sconti per i soci
Programmi di Fidelizzazione e Associativi	• Verifica dell'affiliazione • Verifica della partecipazione • Utilizzo dei benefici fedeltà

continues on next page

Tabella 7.8 – continua dalla pagina precedente

Classe di Credenziale	Finalità Supportate
Documenti di Lavoro	• Verifica dello stato occupazionale • Validazione del profilo professionale
Stato Occupazionale	• Verifica dei contributi previdenziali • Idoneità ai benefici correlati all'occupazione
Patenti e Autorizzazioni	• Verifica dei diritti di guida • Verifica dei diritti di navigazione • Controlli delle forze dell'ordine
Documenti del Veicolo	• Verifica dell'immatricolazione del veicolo • Verifica della revisione del veicolo • Verifica dello stato assicurativo
Abbonamenti al Trasporto	• Accesso ai servizi di trasporto • Verifica dell'abbonamento al trasporto pubblico
Documenti di Viaggio	• Diritto di viaggio o circolazione • Verifica dell'identità per la mobilità transfrontaliera
Assicurazione Viaggio e Prenotazioni	• Verifica della copertura assicurativa di viaggio • Verifica della prenotazione alloggio • Verifica della prenotazione di trasporto
Sconti e Agevolazioni	• Applicazione degli sconti per i soci • Accesso alle agevolazioni turistiche

continues on next page

Tabella 7.8 – continua dalla pagina precedente

Classe di Credenziale	Finalità Supportate
Benefici Economici e Sussidi	• Verifica dell'idoneità ai benefici familiari • Verifica dell'idoneità ai sussidi di disoccupazione • Erogazione del supporto economico
Incentivi e Voucher	• Utilizzo dei voucher digitali • Utilizzo degli incentivi all'acquisto • Verifica dell'idoneità al cashback
Bonus Salute e Benessere	• Accesso ai bonus sanitari • Utilizzo dei voucher per la salute mentale • Utilizzo dei voucher per lo sport
Affiliazione Lavorativa	• Verifica del permesso di accesso

Ogni Credenziale DEVE specificare domini, classi e finalità per abilitare sia gli **Scenari Credential-Specific** che gli **Scenari Credential-Agnostic** in base ai requisiti della Relying Party e ai pattern di richiesta di presentazione, come definito nelle tabelle di mappatura sopra.

1. **Scenari Credential-Specific** (Primari per il settore governativo/ regolamentato): Le RP richiedono tipologie specifiche di Credenziale per requisiti di conformità e audit, inclusi ad esempio:

- **Servizi Governativi**: `"credential_type":"pid"` per la verifica dell'identità specifica con PID oppure `"credential_type":"eid"` per la verifica dell'identità specifica con IT-Wallet ID.
- **Controlli di Polizia**: `"credential_type":"mDL"` per la verifica della patente di guida.
- **KYC Bancario**: Tipologie specifiche di credenziale previste dalla normativa finanziaria.
- **Servizi Sanitari**: `"credential_type":"european_disability_card"` per l'accesso ai benefici per disabilità conformi EU.

2. **Scenari Credential-Agnostic** (Tipici per le aziende private): Le RP richiedono claim specifici indipendentemente dalla fonte della Credenziale per l'efficienza operativa, quali:

- **Consegna E-commerce**: Qualsiasi credenziale, tra quelle a cui è autorizzato ad accedere, contenente `given_name`, `family_name`, `address` per la spedizione.
- **Abbonamenti**: Qualsiasi credenziale, tra quelle a cui è autorizzato ad accedere, con `given_name`, `email` per la personalizzazione.
- **Personalizzazione del Servizio**: Applicazioni aziendali che richiedono dati personali di base senza forti requisiti sulla fonte.

Questo approccio consente:

- **Autorizzazione basata su politiche** tramite mappature **Dominio / Classe / Tipologia di Credenziale / Finalità**.
- **Registrazione flessibile delle RP** a supporto sia delle esigenze di conformità governativa che dei requisiti operativi aziendali.

7.6.2 Struttura del Catalogo degli Attestati Elettronici

Il contenuto del Catalogo delle Credenziali Digitali è protetto in un JWS che contiene i seguenti parametri dell'intestazione JOSE:

Intestazione JOSE	Descrizione	Riferimento
typ	OBBLIGATORIO. DEVE essere impostato su `JOSE`.	[**RFC 7515** Section 4.1. 9].
alg	OBBLIGATORIO. Identificatore di algoritmo di firma digitale come da registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` né con un identificatore di algoritmo simmetrico (MAC).	[**RFC 7515** Section 4.1. 1].
kid	OBBLIGATORIO. Identificatore univoco della chiave pubblica.	[**RFC 7515** Section 4.1. 4].
x5c	OPZIONALE. Contiene il Certificato X.509 della chiave pubblica o la catena di certificati [**RFC 5280**] corrispondente alla chiave utilizzata per firmare digitalmente il JWS. Quando il parametro di intestazione *kid* è presente, DEVE fare riferimento alla stessa chiave pubblica crittografica foglia utilizzata con il Certificato X.509.	[**RFC 7515** Section 4.1. 6.].

continues on next page

Tabella 7.9 – continua dalla pagina precedente

Intestazione JOSE	Descrizione	Riferimento
cty	OBBLIGATORIO. DEVE essere impostato su `application/json`.	[**RFC 7515** Section 4.1. 6.].

Il payload JWS contiene i seguenti parametri:

Tabella 7.10: Campi di Primo Livello del Catalogo delle Credenziali Digitali

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del Catalogo delle Credenziali Digitali (es. `urn:credential-catalog:it-wallet`).
version	OBBLIGATORIO. Versione del formato del Catalogo delle Credenziali Digitali.
last_modified	OBBLIGATORIO. Timestamp dell'ultima modifica al Catalogo delle Credenziali Digitali (es. `2025-03-15T12:00:00Z`).
iss	OBBLIGATORIO. Identificatore dell'emittente del Catalogo delle Credenziali Digitali.
localization	OBBLIGATORIO. Oggetto di configurazione della localizzazione contenente: • **default_locale**: Codice locale predefinito (es. `it`). • **available_locales**: Array dei codici locale supportati (es. `["en", "it"]`). • **base_uri**: URI base per il recupero del bundle di localizzazione (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog/`). • **version**: Versione del formato del bundle di localizzazione.
credentials	OBBLIGATORIO. Array contenente le definizioni delle Credenziali Digitali.

Ogni elemento dell'array `credentials` contiene almeno le seguenti informazioni:

Tabella 7.11: Campi di Primo Livello di Ogni Voce di Credenziale

Nome Campo	Descrizione
version	OBBLIGATORIO. Versione della definizione della Credenziale Digitale.
credential_type	OBBLIGATORIO. Identificatore univoco della tipologia di Credenziale Digitale. Per il PID DEVE essere `pid` e per l'IT-Wallet ID DEVE essere `eid`.

continues on next page

Tabella 7.11 – continua dalla pagina precedente

Nome Campo	Descrizione
credential_name_l10n_id	OBBLIGATORIO. Chiave di localizzazione che fa riferimento al nome leggibile della Credenziale Digitale nel bundle di localizzazione (es. `mDL.name`).
legal_type	OBBLIGATORIO. Classificazione legale della Credenziale (es. `pub-eaa`, `qeaa`, `eaa`).
restriction_policy	OPZIONALE. Restrizioni legali sulle Soluzioni Wallet e/o sugli Emittenti di Credenziali autorizzati a richiedere/emettere la Credenziale Digitale. • **allowed_wallet_ids**: Elenco degli identificatori delle Soluzioni Wallet consentite. • **allowed_issuer_ids**: Elenco degli identificatori degli Emittenti di Credenziali consentiti. Se presente, rappresenta una whitelist di Emittenti di Credenziali che possono essere aggiunti dal Trust Anchor nel campo **issuers** della corrispondente Credenziale Digitale. • **presentation_flows**: Tipo di flussi di presentazione supportati; flusso remoto e/o di prossimità.
pricing_policy	OPZIONALE. Informazioni sul prezzo della Credenziale Digitale, inclusi: • **models**: OBBLIGATORIO. Array di modelli di prezzo applicabili alla Credenziale Digitale, ciascuno contenente – **pricing_type**: Tipo di modello di prezzo, come `issuance_based`, `verification_based`, `subscription_based`, `other`. – **price**: Costo associato al modello. – **currency**: Valuta del prezzo. • **pricing_model_uri**: URI alla documentazione dettagliata del modello di prezzo.

continues on next page

Tabella 7.11 – continua dalla pagina precedente

Nome Campo	Descrizione
validity_info	OBBLIGATORIO. Informazioni sulla validità della Credenziale Digitale, inclusi almeno: • **max_validity_days**: Periodo massimo di validità in giorni. • **status_methods**: Metodi di verifica dello stato supportati (es. `status_list`). • **allowed_states**: Array di oggetti che rappresentano gli stati consentiti della Credenziale Digitale. Ogni oggetto contiene un codice di stato esadecimale (es. `0x00` per `VALID`, `0x01` per `INVALID`, `0x02` per `SUSPENDED`, `0x03` per `UPDATE`, `0x0F` per `ATTRIBUTE_UPDATE`), una chiave di localizzazione `title_l10n_id` e una chiave di localizzazione `description_l10n_id` per la visualizzazione nell'interfaccia utente. • **administrative_expiration_user_info**: OPZIONALE. Oggetto contenente chiavi `title_l10n_id` e `description_l10n_id` per visualizzare le informazioni sulla scadenza amministrativa all'Utente.
authentication	OBBLIGATORIO. Requisiti di autenticazione della Credenziale Digitale. • **user_auth_required**: OBBLIGATORIO. Flag che indica se l'autenticazione dell'Utente è richiesta durante l'emissione della Credenziale Digitale. • **min_loa**: OBBLIGATORIO. Livello Minimo di Garanzia richiesto per l'autenticazione della Credenziale Digitale. DEVE includere il Livello di Garanzia dell'autenticazione dell'Utente e dell'Istanza Wallet che richiede la Credenziale Digitale. • **supported_schemes**: OBBLIGATORIO se `user_auth_required` è `true`. Schemi di autenticazione dell'identità digitale supportati (es. `["it_wallet"]`).
domains	OBBLIGATORIO. Array contenente gli ID dei domini a cui appartiene la Credenziale Digitale (es. `"IDENTITY"`, `"MOBILITY_TRAVEL"`).
classes	OBBLIGATORIO. Array contenente gli ID delle classi a cui appartiene la Credenziale Digitale (es. `"IDENTIFICATION_DOCUMENTS"`, `"LICENSES_AUTHORIZATIONS"`).
purposes	OBBLIGATORIO. Array contenente gli ID delle finalità (definita nella Tassonomia) di utilizzo per cui la Credenziale Digitale può essere impiegata, definendo contesti d'uso specifici e claim richiesti per ciascuna finalità (es. `"IDENTITY_VERIFICATION"`, `"AGE_VERIFICATION"`, `"DRIVING_RIGHTS_VERIFICATION"`).

continues on next page

Tabella 7.11 – continua dalla pagina precedente

Nome Campo	Descrizione
issuers	OBBLIGATORIO. Array di informazioni rilevanti sugli Emittenti di Credenziali autorizzati, inclusi dati amministrativi e tecnici quali nome dell'organizzazione, riferimento al documento di specifica API e meccanismi di emissione supportati. Ogni elemento dell'array contiene: • **entity_id**: OBBLIGATORIO. Stringa. Identificativo univoco del Credential Issuer. DEVE corrispondere al valore contenuto nel parametro `iss` dell'Entity Configuration del Credential Issuer. • **organization_name_l10n_id**: OBBLIGATORIO. Stringa. Chiave di localizzazione che fa riferimento al nome localizzato dell'organizzazione nel bundle di localizzazione (es. `issuer1.name`). • **organization_code**: OBBLIGATORIO. Stringa. Codice IPA del Credential Issuer per enti pubblici oppure partita IVA per soggetti privati. • **organization_country**: OBBLIGATORIO. Stringa. Codice paese dell'organizzazione a due lettere conforme allo standard ISO 3166-1 alpha-2. • **contacts**: OBBLIGATORIO. Stringa. Array di indirizzi email di contatto per almeno un referente di supporto utenti, uno applicativo e uno specialista di sistemi. • **legal_type**: OBBLIGATORIO. Stringa. Classificazione giuridica del Credential Issuer (es. pub-eaa, qeaa, eaa). • **homepage_uri**: OBBLIGATORIO. Stringa. URL che punta alla homepage dell'organizzazione. • **logo_uri**: OPZIONALE. Stringa. URL dell'immagine del logo dell'organizzazione. • **policy_uri**: OBBLIGATORIO. Stringa. URL al documento di privacy policy. • **tos_uri**: OPZIONALE. Stringa. URL al documento di termini di servizio. • **service_documentation_uri**: OPZIONALE. Stringa. URL che punta alla documentazione del servizio del Credential Issuer. • **issuance_flows**: OBBLIGATORIO. Oggetto. Contiene i seguenti parametri: – **deferred_flow**: OBBLIGATORIO. Booleano. Indica se è supportata l'emissione differita. – **immediate_flows**: OBBLIGATORIO. Booleano. Indica se è supportata l'emissione immediata. – **wallet_initiated**: OBBLIGATORIO. Booleano. Indica se è supportato il flusso Wallet-Initiated. – **issuer_initiated**: OBBLIGATORIO. Booleano. Indica se è supportata il flusso Issuer-Initiated (Third Party Initiated Flow). – **max_deferred_issuance_time_minutes**: CONDIZIONALE. Intero. Tempo massimo, in minuti,

Tabella 7.11 – continua dalla pagina precedente

Nome Campo	Descrizione
authentic_sources	CONDIZIONALE. È OBBLIGATORIO solo se `parent_credentials` è assente. Array di oggetti JSON delle Fonti Autentiche che fanno riferimento alle Fonti Autentiche autorizzate. Ogni oggetto DEVE contenere l'identificatore dell'entità FA e l'identificatore specifico della capacità dati: • **id**: Identificatore stringa che fa riferimento all'entity_id della Fonte Autentica come registrato in *Registro delle Fonti Autentiche* (pagina 173). • **dataset_id**: Identificatore stringa della specifica capacità dati/ dataset utilizzato dall'Emittente dalla FA.
parent_credentials	CONDIZIONALE. È OBBLIGATORIO solo se `authentic_sources` è assente. Array degli identificativi di `credential_type` relativi a Credenziali designate come fonti dei dati. Ogni elemento identifica una Credenziale che funge da Fonte Autentica durante il processo di emissione della Credenziale Digitale.

Nota

L'unione di `credential_type` e `version` DEVE essere univoca nel Catalogo delle Credenziali.

Il corrispondente esempio di Catalogo delle Credenziali Digitali decodificato in JSON sia per l'intestazione che per il payload è il seguente:

```
{
    "typ":"JOSE",
    "alg":"ES256",
    "kid":"e9bc097a-ce51-4036-9562-d2ade882db0d",
    "cty":"application/json"
}
```

```
{
    "id": "urn:credential-catalog:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "iss": "https://trust-registry.eid-wallet.example.it",
    "localization": {
        "available_locales": [
            "en",
            "it"
        ],
```

```
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog",
        "default_locale": "it",
        "version": "1.0.0"
    },
    "credentials": [
        {
            "version": "1",
            "credential_type": "mDL",
            "credential_name_l10n_id": "mDL.name",
            "legal_type": "pub-eaa",
            "restriction_policy": {
                "allowed_wallet_ids": [
                    "https://wallet-provider.example.org/wallet_solution",
                    "https://wallet-provider2.example.org/wallet_solution"
                ],
                "allowed_issuer_ids": [
                    "https://issuer.example.org"
                ],
                "presentation_flows": {
                    "remote": true,
                    "proximity": true
                }
            },
            "pricing_policy": {
                "models": [
                    {
                        "pricing_type": "verification_based",
                        "price": 0.01,
                        "currency": "EUR"
                    }
                ],
                "pricing_model_uri": "https://example.com/pricing"
            },
            "validity_info": {
                "max_validity_days": 365,
                "status_methods": [
                    "status_list"
                ],
                "administrative_expiration_user_info": {
                    "title_l10n_id": "mDL.administrative_expiration_user_info.title",
                    "description_l10n_id": "mDL.administrative_expiration_user_info.description"

                },
                "allowed_states": [
                    {
                    "0x00": "VALID",
```

```
            "title_l10n_id": "mDL.VALID.title",
            "description_l10n_id": "mDL.VALID.description"
            },
            {
            "0x01": "INVALID",
            "title_l10n_id": "mDL.INVALID.title",
            "description_l10n_id": "mDL.INVALID.description"
            },
            {
            "0x02": "SUSPENDED",
            "title_l10n_id": "mDL.SUSPENDED.title",
            "description_l10n_id": "mDL.SUSPENDED.description"
            },
            {
            "0x03": "UPDATE",
            "title_l10n_id": "mDL.UPDATE.title",
            "description_l10n_id": "mDL.UPDATE.description"
            },
            {
            "0x0F": "ATTRIBUTE_UPDATE",
            "title_l10n_id": "mDL.ATTRIBUTE_UPDATE.title",
            "description_l10n_id": "mDL.ATTRIBUTE_UPDATE.description"
            }
        ]
    },
    "authentication": {
        "user_auth_required": true,
        "min_loa": "high",
        "supported_schemes": [
            "it_wallet"
        ]
    },
    "domains": ["IDENTITY", "MOBILITY_TRAVEL"],
    "classes": ["IDENTIFICATION_DOCUMENTS", "LICENSES_AUTHORIZATIONS"],
    "purposes": ["IDENTITY_VERIFICATION", "AGE_VERIFICATION", "DRIVING_RIGHTS"],
    "issuers": [
        {
            "entity_id": "https://issuer.example.org",
            "organization_name_l10n_id": "mDL.issuer1.name",
            "organization_code": "ci_example_it",
            "organization_country": "IT",
            "contacts": [
                "mailto:informazioni@example.it",
                "mailto:protocollo@pec.example.it"
            ],
            "legal_type": "pub-eaa",
            "homepage_uri": "https://issuer.example.org",
```

```
                    "logo_uri": "https://issuer.example.org/logo.svg",
                    "policy_uri": "https://issuer.example.org/privacy",
                    "tos_uri": "https://issuer.example.org/terms",
                    "service_documentation_uri": "https://issuer.example.org/.well-known/servic
e-doc",
                    "issuance_flows": {
                        "wallet_initiated": true,
                        "issuer_initiated": true,
                        "deferred_flow": true,
                        "max_deferred_issuance_time_minutes": 1440,
                        "notification_methods": [
                            "push",
                            "polling"
                        ]
                    }
                }
            ],
            "authentic_sources": [
                {
                    "id": "https://motorizzazione.gov.example",
                    "dataset_id": "38832801"
                }
            ]
        }
    ]
}
```


Nota

Per una gestione migliore e più efficiente della localizzazione delle informazioni contenute nel Catalogo delle Credenziali Digitali, un'Entità che lo consulta DOVREBBE:

- Scaricare la versione base del Catalogo delle Credenziali Digitali (compatta, senza localizzazioni) utilizzando l'endpoint `.well-known/credential-catalog`.
- Determinare la lingua preferita dall'Utente.
- Scaricare solo i bundle di localizzazione necessari.
- Unire dinamicamente il contenuto localizzato con la Struttura del Catalogo degli Attestati Elettronici.

Di seguito è fornito un esempio non normativo dell'output di un bundle di localizzazione:


```
{
  "mDL.name": "Patente di Guida",
  "mDL.issuer1.name": "Esempio di Credential Issuer",
```

```
  "...": "..."
}
```

I bundle di localizzazione DEVONO essere disponibili all'URI composto aggiungendo il codice locale e `.json` al valore `localization.base_uri` definito nel catalogo. Ciascun bundle locale DEVE essere accessibile seguendo il pattern di denominazione **{locale_code}.json**, dove **{locale_code}** è sostituito con il codice locale corrispondente dall'array **available_locales**.

Un esempio non normativo dell'URI di localizzazione italiana per il bundle sarebbe **https://trust-registry.eid-wallet.example.it/.well-known/l10n/credential-catalog/it.json**.

7.6.3 Decentralizzazione delle Informazioni di Visualizzazione e dei Claims

La fonte canonica per le caratteristiche di visualizzazione e la struttura dei claim è determinata dai **Metadati dell'Emittente di Credenziali (Configurazione dell'Entità)**.

La logica complessiva per la presentazione di una Credenziale è:

1. Il Wallet/Relying Party recupera il **Catalogo delle Credenziali Digitali** leggero per scoprire la `credential_type` disponibile e l'`entity_id` dei relativi Emittenti di Credenziali.
2. Recupera i **Metadati dell'Emittente di Credenziali** completi (Configurazione dell'Entità) dall'`entity_id` rilevato.
3. I Metadati dell'Emittente di Credenziali DEVONO contenere le caratteristiche di visualizzazione complete (logo, colori) e le informazioni dettagliate sullo schema (tramite link ai Metadati di Tipo appropriati o direttamente nella configurazione). L'Emittente costruisce questi metadati sulla base dei suggerimenti forniti dalla Fonte Autentica (tramite il Registro FA) e delle specifiche di schema standard (tramite il Registro degli Schemi).

7.7 Tassonomia

La **Tassonomia** fornisce il fondamento semantico per l'interoperabilità delle Credenziali Digitali mantenendo il vocabolario autorevole per l'organizzazione delle Credenziali all'interno dell'ecosistema IT-Wallet. La tassonomia è neutrale rispetto al formato della Credenziale.

La Tassonomia fornisce, in un'unica risorsa, il sistema di classificazione gerarchica che organizza Domini, Classi e Finalità applicabili alle Tipologie di Credenziale, supportando la valutazione delle politiche di autorizzazione e la standardizzazione a livello di ecosistema.

Obiettivi della Tassonomia:

1. **Fondamento Semantico**: Stabilire un vocabolario standardizzato per domini e finalità nell'intero ecosistema

2. **Framework per le Politiche**: Abilitare decisioni di autorizzazione strutturate basate sulla classificazione gerarchica
3. **Interoperabilità**: Garantire un'interpretazione coerente delle classificazioni delle credenziali
4. **Estensibilità**: Supportare l'evoluzione dell'ecosistema con nuovi Domini, Classi, Tipologie di Credenziale e Finalità
5. **Conformità Transfrontaliera**: Allineamento con i requisiti normativi EU e gli standard internazionali

Struttura della Tassonomia:

La tassonomia mantiene una struttura gerarchica a quattro livelli:

- **Domini**: Classificazione di primo livello che rappresenta ampie aree funzionali (es. IDENTITY, HEALTH, FINANCIAL, AUTHENTICATION)
- **Classe (Famiglia di Credenziali)**: Famiglia di Credenziali che condividono funzione, struttura o significato giuridico simili (es. Documenti di Identità, Certificati di Stato Civile, Albi Professionali, Accesso)
- **Tipologia di Credenziale**: Definizione specifica di Credenziale emessa da un'autorità (es. Credenziale di Viaggio Digitale, Certificato di Nascita, Patente di Guida Mobile).
- **Finalità (Intento di Verifica)**: Obiettivi di verifica che una Credenziale può soddisfare (es. Verifica dell'Identità, Verifica dell'Età, Idoneità a servizi specifici, Verifica del permesso di accesso).

Nota

La Tipologia di Credenziale è un concetto definito a livello del Catalogo delle Credenziali Digitali, non all'interno della Tassonomia. La Tassonomia fornisce il vocabolario di classificazione (Domini, Classi, Finalità) a cui le Tipologie di Credenziale nel Catalogo fanno riferimento.

Supporto alla Localizzazione:

La tassonomia supporta ambienti multilingue tramite il pattern con suffisso `_l10n_id`, abilitando una gestione efficiente della localizzazione per le interfacce utente e le implementazioni transfrontaliere.

Utilizzo della Tassonomia:

- **Registro dei Claims**: Catalogo dei singoli claim
- **Registro FA**: Le Fonti Autentiche dichiarano le capacità utilizzando le classificazioni della tassonomia

- **Catalogo delle Credenziali Digitali**: Le Tipologie di Credenziale specificano Domini, Classi e Finalità
- **Politiche di Autorizzazione**: La valutazione delle politiche sfrutta la struttura della tassonomia per le decisioni di controllo degli accessi

La Tassonomia è accessibile attraverso l'endpoint dedicato alla tassonomia come definito nel meccanismo di discovery del registro ed è mantenuta dall'Organismo di Vigilanza per garantire la conformità normativa e la coerenza semantica.

Struttura JSON della Tassonomia:

Tabella 7.12: Campi di Primo Livello della Tassonomia

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco della Tassonomia (es. `urn:taxonomy:it-wallet`).
version	OBBLIGATORIO. La versione della Tassonomia (es. `1.0.0`).
last_modified	OBBLIGATORIO. Il timestamp che indica quando la Tassonomia è stata aggiornata l'ultima volta (es. `2026-03-11T00:00:00Z`).
name_l10n_id	OBBLIGATORIO. Chiave di localizzazione che fa riferimento al nome leggibile della Tassonomia (es. `taxonomy.name`).
description_l10n_id	OBBLIGATORIO. Chiave di localizzazione che fa riferimento alla descrizione leggibile della Tassonomia (es. `taxonomy.description`).
localization	OBBLIGATORIO. Oggetto di configurazione della localizzazione contenente: • **default_locale**: Codice locale predefinito (es. `it`). • **available_locales**: Array dei codici locale supportati (es. `["en", "it"]`). • **base_uri**: URI base per il recupero del bundle di localizzazione (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/`). • **version**: Versione del formato del bundle di localizzazione.

continues on next page

Tabella 7.12 – continua dalla pagina precedente

Nome Campo	Descrizione
domains	OBBLIGATORIO. Array di oggetti Dominio, ciascuno contenente: • **id**: Identificatore univoco del Dominio in SCREAMING_SNAKE_CASE (es. `IDENTITY`). • **name_l10n_id**: Chiave di localizzazione per il nome del dominio (es. `domain.identity.name`). • **description_l10n_id**: Chiave di localizzazione per la descrizione del dominio (es. `domain.identity.description`). • **classes**: Array di oggetti Classe. Ogni classe contiene `id`, `name_l10n_id` e `supported_purposes` (array di stringhe di ID finalità).
purposes	OBBLIGATORIO. Array piatto di tutti gli oggetti Finalità definiti nella tassonomia, ciascuno contenente: • **id**: Identificatore univoco della Finalità in SCREAMING_SNAKE_CASE (es. `IDENTITY_VERIFICATION`, `ACCESS_PERMIT`). • **name_l10n_id**: Chiave di localizzazione per il nome della finalità (es. `purpose.identity_verification.name`).

Di seguito è fornito un esempio non normativo della struttura della Tassonomia:


```
{
    "id": "urn:taxonomy:it-wallet",

    "version": "1.0.0",
    "last_modified": "2026-03-11T00:00:00Z",
    "name_l10n_id": "taxonomy.name",
    "description_l10n_id": "taxonomy.description",
    "localization": {
        "available_locales": [
            "en",
            "it"
        ],
        "base_uri": "https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/",
        "default_locale": "it",
        "version": "1.0.0"
    },
    "domains": [
        {
            "classes": [
                {
                    "id": "IDENTIFICATION_DOCUMENTS",
                    "name_l10n_id": "class.identification_documents.name",
```

```
                "supported_purposes": [
                    "IDENTITY_VERIFICATION",
                    "AGE_VERIFICATION",
                    "PERSON_IDENTIFICATION"
                ]
            },
            {
                "id": "CIVIL_REGISTRY_STATUS",
                "name_l10n_id": "class.civil_registry_status.name",
                "supported_purposes": [
                    "CIVIL_STATUS_VERIFICATION",
                    "RIGHT_OF_RESIDENCE",
                    "HOUSEHOLD_COMPOSITION_VERIFICATION"
                ]
            },
            {
                "id": "ECONOMIC_LEGAL_STATUS",
                "name_l10n_id": "class.economic_legal_status.name",
                "supported_purposes": [
                    "ELIGIBILITY_VERIFICATION",
                    "LEGAL_STATUS_VERIFICATION",
                    "CRIMINAL_RECORD_CHECK"
                ]
            }
        ],
        "description_l10n_id": "domain.identity.description",
        "id": "IDENTITY",
        "name_l10n_id": "domain.identity.name"
    },
    {
        "classes": [
            {
                "id": "PROPERTY_CADASTRAL_DOCS",
                "name_l10n_id": "class.property_cadastral_docs.name",
                "supported_purposes": [
                    "RESIDENCE_HOUSEHOLD_VERIFICATION",
                    "PROPERTY_OWNERSHIP_VERIFICATION",
                    "REAL_ESTATE_COMPLIANCE"
                ]
            },
            {
                "id": "FAMILY_DOCUMENTS",
                "name_l10n_id": "class.family_documents.name",
                "supported_purposes": [
                    "HOUSEHOLD_COMPOSITION_VERIFICATION",
                    "FAMILY_SOCIAL_SERVICES_ELIGIBILITY"
                ]
```

```
            },
            {
                "id": "LOCAL_TAX_DOCUMENTS",
                "name_l10n_id": "class.local_tax_documents.name",
                "supported_purposes": [
                    "LOCAL_TAX_COMPLIANCE",
                    "PROPERTY_TAX_VERIFICATION"
                ]
            }
        ],
        "description_l10n_id": "domain.home_family.description",
        "id": "HOME_FAMILY",
        "name_l10n_id": "domain.home_family.name"
    },
    {
        "classes": [
            {
                "id": "EDUCATIONAL_QUALIFICATIONS",
                "name_l10n_id": "class.educational_qualifications.name",
                "supported_purposes": [
                    "QUALIFICATION_DEGREE_VERIFICATION",
                    "EDUCATION_PATHWAY_ELIGIBILITY"
                ]
            },
            {
                "id": "PROFESSIONAL_CERTIFICATIONS",
                "name_l10n_id": "class.professional_certifications.name",
                "supported_purposes": [
                    "PROFESSIONAL_LICENSE_VERIFICATION",
                    "SKILLS_ASSESSMENT"
                ]
            }
        ],
        "description_l10n_id": "domain.education.description",
        "id": "EDUCATION",
        "name_l10n_id": "domain.education.name"
    },
    {
        "classes": [
            {
                "id": "CERTIFICATIONS_ELIGIBILITY",
                "name_l10n_id": "class.certifications_eligibility.name",
                "supported_purposes": [
                    "VACCINATION_STATUS_VERIFICATION",
                    "FITNESS_STATUS_VERIFICATION",
                    "HEALTH_RESTRICTED_ACCESS"
                ]
```

```
            },
            {
                "id": "MEDICAL_RECORDS",
                "name_l10n_id": "class.medical_records.name",
                "supported_purposes": [
                    "HEALTHCARE_SERVICES_ACCESS",
                    "MEDICAL_RECORDS_SHARING",
                    "MEDICAL_HISTORY_VALIDATION"
                ]
            }
        ],
        "description_l10n_id": "domain.health.description",
        "id": "HEALTH",
        "name_l10n_id": "domain.health.name"
    },
    {
        "classes": [
            {
                "id": "PAYMENT_INSTRUMENTS",
                "name_l10n_id": "class.payment_instruments.name",
                "supported_purposes": [
                    "PAYMENT_AUTHORIZATION",
                    "PAYMENT_EXECUTION",
                    "PROOF_OF_PAYMENT"
                ]
            },
            {
                "id": "PAYMENT_CREDENTIALS_AUTHORIZATIONS",
                "name_l10n_id": "class.payment_credentials_authorizations.name",
                "supported_purposes": [
                    "FINANCIAL_AUTHORIZATION_MANAGEMENT",
                    "STRONG_CUSTOMER_AUTHENTICATION"
                ]
            },
            {
                "id": "PUBLIC_PAYMENTS_FEES",
                "name_l10n_id": "class.public_payments_fees.name",
                "supported_purposes": [
                    "PROOF_OF_TAX_PAYMENT",
                    "PROOF_OF_FEE_PAYMENT",
                    "DIGITAL_STAMP_DUTY_VALIDATION"
                ]
            },
            {
                "id": "RECURRING_PAYMENTS_SUBSCRIPTIONS",
                "name_l10n_id": "class.recurring_payments_subscriptions.name",
                "supported_purposes": [
```

```
                "RECURRING_PAYMENT_MANAGEMENT",
                "SUBSCRIPTION_MANDATE_VERIFICATION"
            ]
        }
    ],
    "description_l10n_id": "domain.financial.description",
    "id": "FINANCIAL",
    "name_l10n_id": "domain.financial.name"
},
{
    "classes": [
        {
            "id": "CULTURAL_CARDS_BENEFITS",
            "name_l10n_id": "class.cultural_cards_benefits.name",
            "supported_purposes": [
                "CULTURAL_SERVICES_ACCESS",
                "LEISURE_SERVICES_ACCESS",
                "MEMBER_DISCOUNT_APPLICATION"
            ]
        },
        {
            "id": "MEMBERSHIP_LOYALTY_PROGRAMS",
            "name_l10n_id": "class.membership_loyalty_programs.name",
            "supported_purposes": [
                "AFFILIATION_VERIFICATION",
                "PARTICIPATION_VERIFICATION",
                "LOYALTY_BENEFITS_USE"
            ]
        }
    ],
    "description_l10n_id": "domain.culture_leisure.description",
    "id": "CULTURE_LEISURE",
    "name_l10n_id": "domain.culture_leisure.name"
},
{
    "classes": [
        {
            "id": "EMPLOYMENT_DOCUMENTS",
            "name_l10n_id": "class.employment_documents.name",
            "supported_purposes": [
                "EMPLOYMENT_STATUS_VERIFICATION",
                "PROFESSIONAL_PROFILE_VALIDATION"
            ]
        },
        {
            "id": "EMPLOYMENT_STATUS",
            "name_l10n_id": "class.employment_status.name",
```

```
                "supported_purposes": [
                    "CONTRIBUTION_RECORDS_VERIFICATION",
                    "EMPLOYMENT_BENEFITS_ELIGIBILITY"
                ]
            },
            {
                "id": "EMPLOYMENT_AFFILIATION",
                "name_l10n_id": "class.employment_affiliation.name",
                "supported_purposes": [
                    "ACCESS_PERMIT"
                ]
            }
        ],
        "description_l10n_id": "domain.employment.description",
        "id": "EMPLOYMENT",
        "name_l10n_id": "domain.employment.name"
    },
    {
        "classes": [
            {
                "id": "LICENSES_AUTHORIZATIONS",
                "name_l10n_id": "class.licenses_authorizations.name",
                "supported_purposes": [
                    "DRIVING_RIGHTS_VERIFICATION",
                    "NAVIGATION_RIGHTS_VERIFICATION",
                    "LAW_ENFORCEMENT_CONTROLS"
                ]
            },
            {
                "id": "VEHICLE_DOCUMENTS",
                "name_l10n_id": "class.vehicle_documents.name",
                "supported_purposes": [
                    "VEHICLE_REGISTRATION_VERIFICATION",
                    "VEHICLE_INSPECTION_VERIFICATION",
                    "INSURANCE_STATUS_CHECK"
                ]
            },
            {
                "id": "TRANSPORT_SUBSCRIPTIONS",
                "name_l10n_id": "class.transport_subscriptions.name",
                "supported_purposes": [
                    "TRANSPORT_SERVICES_ACCESS",
                    "PUBLIC_TRANSPORT_PASS_VERIFICATION"
                ]
            },
            {
                "id": "TRAVEL_DOCUMENTS",
```

```
                "name_l10n_id": "class.travel_documents.name",
                "supported_purposes": [
                    "RIGHT_TO_TRAVEL",
                    "CROSS_BORDER_IDENTITY_CHECK"
                ]
            },
            {
                "id": "TRAVEL_INSURANCE_BOOKINGS",
                "name_l10n_id": "class.travel_insurance_bookings.name",
                "supported_purposes": [
                    "TRAVEL_INSURANCE_VERIFICATION",
                    "ACCOMMODATION_RESERVATION_CHECK",
                    "TRANSPORT_RESERVATION_CHECK"
                ]
            },
            {
                "id": "DISCOUNTS_BENEFITS",
                "name_l10n_id": "class.discounts_benefits.name",
                "supported_purposes": [
                    "MEMBER_DISCOUNT_APPLICATION",
                    "TOURIST_BENEFITS_ACCESS"
                ]
            }
        ],
        "description_l10n_id": "domain.mobility_travel.description",
        "id": "MOBILITY_TRAVEL",
        "name_l10n_id": "domain.mobility_travel.name"
    },
    {
        "classes": [
            {
                "id": "ECONOMIC_BENEFITS_ALLOWANCES",
                "name_l10n_id": "class.economic_benefits_allowances.name",
                "supported_purposes": [
                    "FAMILY_BENEFITS_ELIGIBILITY",
                    "UNEMPLOYMENT_BENEFITS_ELIGIBILITY",
                    "ECONOMIC_SUPPORT_ALLOCATION"
                ]
            },
            {
                "id": "INCENTIVES_VOUCHERS",
                "name_l10n_id": "class.incentives_vouchers.name",
                "supported_purposes": [
                    "DIGITAL_VOUCHER_USE",
                    "PURCHASE_INCENTIVE_USE",
                    "CASHBACK_ELIGIBILITY_VERIFICATION"
                ]
```

```
            },
            {
                "id": "HEALTH_WELLBEING_BONUSES",
                "name_l10n_id": "class.health_wellbeing_bonuses.name",
                "supported_purposes": [
                    "HEALTHCARE_BONUS_ACCESS",
                    "MENTAL_HEALTH_VOUCHER_USE",
                    "SPORTS_VOUCHER_USE"
                ]
            }
        ],
        "description_l10n_id": "domain.bonuses.description",
        "id": "BONUSES",
        "name_l10n_id": "domain.bonuses.name"
    }
],
"purposes": [
    {
        "id": "IDENTITY_VERIFICATION",
        "name_l10n_id": "purpose.identity_verification.name"
    },
    {
        "id": "AGE_VERIFICATION",
        "name_l10n_id": "purpose.age_verification.name"
    },
    {
        "id": "PERSON_IDENTIFICATION",
        "name_l10n_id": "purpose.person_identification.name"
    },
    {
        "id": "CIVIL_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.civil_status_verification.name"
    },
    {
        "id": "RIGHT_OF_RESIDENCE",
        "name_l10n_id": "purpose.right_of_residence.name"
    },
    {
        "id": "HOUSEHOLD_COMPOSITION_VERIFICATION",
        "name_l10n_id": "purpose.household_composition_verification.name"
    },
    {
        "id": "ELIGIBILITY_VERIFICATION",
        "name_l10n_id": "purpose.eligibility_verification.name"
    },
    {
        "id": "LEGAL_STATUS_VERIFICATION",
```

```
        "name_l10n_id": "purpose.legal_status_verification.name"
    },
    {
        "id": "CRIMINAL_RECORD_CHECK",
        "name_l10n_id": "purpose.criminal_record_check.name"
    },
    {
        "id": "RESIDENCE_HOUSEHOLD_VERIFICATION",
        "name_l10n_id": "purpose.residence_household_verification.name"
    },
    {
        "id": "PROPERTY_OWNERSHIP_VERIFICATION",
        "name_l10n_id": "purpose.property_ownership_verification.name"
    },
    {
        "id": "REAL_ESTATE_COMPLIANCE",
        "name_l10n_id": "purpose.real_estate_compliance.name"
    },
    {
        "id": "FAMILY_SOCIAL_SERVICES_ELIGIBILITY",
        "name_l10n_id": "purpose.family_social_services_eligibility.name"
    },
    {
        "id": "LOCAL_TAX_COMPLIANCE",
        "name_l10n_id": "purpose.local_tax_compliance.name"
    },
    {
        "id": "PROPERTY_TAX_VERIFICATION",
        "name_l10n_id": "purpose.property_tax_verification.name"
    },
    {
        "id": "QUALIFICATION_DEGREE_VERIFICATION",
        "name_l10n_id": "purpose.qualification_degree_verification.name"
    },
    {
        "id": "EDUCATION_PATHWAY_ELIGIBILITY",
        "name_l10n_id": "purpose.education_pathway_eligibility.name"
    },
    {
        "id": "PROFESSIONAL_LICENSE_VERIFICATION",
        "name_l10n_id": "purpose.professional_license_verification.name"
    },
    {
        "id": "SKILLS_ASSESSMENT",
        "name_l10n_id": "purpose.skills_assessment.name"
    },
    {
```

```
        "id": "VACCINATION_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.vaccination_status_verification.name"
    },
    {
        "id": "FITNESS_STATUS_VERIFICATION",
        "name_l10n_id": "purpose.fitness_status_verification.name"
    },
    {
        "id": "HEALTH_RESTRICTED_ACCESS",
        "name_l10n_id": "purpose.health_restricted_access.name"
    },
    {
        "id": "HEALTHCARE_SERVICES_ACCESS",
        "name_l10n_id": "purpose.healthcare_services_access.name"
    },
    {
        "id": "MEDICAL_RECORDS_SHARING",
        "name_l10n_id": "purpose.medical_records_sharing.name"
    },
    {
        "id": "MEDICAL_HISTORY_VALIDATION",
        "name_l10n_id": "purpose.medical_history_validation.name"
    },
    {
        "id": "PAYMENT_AUTHORIZATION",
        "name_l10n_id": "purpose.payment_authorization.name"
    },
    {
        "id": "PAYMENT_EXECUTION",
        "name_l10n_id": "purpose.payment_execution.name"
    },
    {
        "id": "PROOF_OF_PAYMENT",
        "name_l10n_id": "purpose.proof_of_payment.name"
    },
    {
        "id": "FINANCIAL_AUTHORIZATION_MANAGEMENT",
        "name_l10n_id": "purpose.financial_authorization_management.name"
    },
    {
        "id": "STRONG_CUSTOMER_AUTHENTICATION",
        "name_l10n_id": "purpose.strong_customer_authentication.name"
    },
    {
        "id": "PROOF_OF_TAX_PAYMENT",
        "name_l10n_id": "purpose.proof_of_tax_payment.name"
    },
```

```
{
    "id": "PROOF_OF_FEE_PAYMENT",
    "name_l10n_id": "purpose.proof_of_fee_payment.name"
},
{
    "id": "DIGITAL_STAMP_DUTY_VALIDATION",
    "name_l10n_id": "purpose.digital_stamp_duty_validation.name"
},
{
    "id": "RECURRING_PAYMENT_MANAGEMENT",
    "name_l10n_id": "purpose.recurring_payment_management.name"
},
{
    "id": "SUBSCRIPTION_MANDATE_VERIFICATION",
    "name_l10n_id": "purpose.subscription_mandate_verification.name"
},
{
    "id": "CULTURAL_SERVICES_ACCESS",
    "name_l10n_id": "purpose.cultural_services_access.name"
},
{
    "id": "LEISURE_SERVICES_ACCESS",
    "name_l10n_id": "purpose.leisure_services_access.name"
},
{
    "id": "MEMBER_DISCOUNT_APPLICATION",
    "name_l10n_id": "purpose.member_discount_application.name"
},
{
    "id": "AFFILIATION_VERIFICATION",
    "name_l10n_id": "purpose.affiliation_verification.name"
},
{
    "id": "PARTICIPATION_VERIFICATION",
    "name_l10n_id": "purpose.participation_verification.name"
},
{
    "id": "LOYALTY_BENEFITS_USE",
    "name_l10n_id": "purpose.loyalty_benefits_use.name"
},
{
    "id": "EMPLOYMENT_STATUS_VERIFICATION",
    "name_l10n_id": "purpose.employment_status_verification.name"
},
{
    "id": "PROFESSIONAL_PROFILE_VALIDATION",
    "name_l10n_id": "purpose.professional_profile_validation.name"
```

```
},
{
    "id": "CONTRIBUTION_RECORDS_VERIFICATION",
    "name_l10n_id": "purpose.contribution_records_verification.name"
},
{
    "id": "EMPLOYMENT_BENEFITS_ELIGIBILITY",
    "name_l10n_id": "purpose.employment_benefits_eligibility.name"
},
{
    "id": "DRIVING_RIGHTS_VERIFICATION",
    "name_l10n_id": "purpose.driving_rights_verification.name"
},
{
    "id": "NAVIGATION_RIGHTS_VERIFICATION",
    "name_l10n_id": "purpose.navigation_rights_verification.name"
},
{
    "id": "LAW_ENFORCEMENT_CONTROLS",
    "name_l10n_id": "purpose.law_enforcement_controls.name"
},
{
    "id": "VEHICLE_REGISTRATION_VERIFICATION",
    "name_l10n_id": "purpose.vehicle_registration_verification.name"
},
{
    "id": "VEHICLE_INSPECTION_VERIFICATION",
    "name_l10n_id": "purpose.vehicle_inspection_verification.name"
},
{
    "id": "INSURANCE_STATUS_CHECK",
    "name_l10n_id": "purpose.insurance_status_check.name"
},
{
    "id": "TRANSPORT_SERVICES_ACCESS",
    "name_l10n_id": "purpose.transport_services_access.name"
},
{
    "id": "PUBLIC_TRANSPORT_PASS_VERIFICATION",
    "name_l10n_id": "purpose.public_transport_pass_verification.name"
},
{
    "id": "RIGHT_TO_TRAVEL",
    "name_l10n_id": "purpose.right_to_travel.name"
},
{
    "id": "CROSS_BORDER_IDENTITY_CHECK",
```

```
        "name_l10n_id": "purpose.cross_border_identity_check.name"
    },
    {
        "id": "TRAVEL_INSURANCE_VERIFICATION",
        "name_l10n_id": "purpose.travel_insurance_verification.name"
    },
    {
        "id": "ACCOMMODATION_RESERVATION_CHECK",
        "name_l10n_id": "purpose.accommodation_reservation_check.name"
    },
    {
        "id": "TRANSPORT_RESERVATION_CHECK",
        "name_l10n_id": "purpose.transport_reservation_check.name"
    },
    {
        "id": "TOURIST_BENEFITS_ACCESS",
        "name_l10n_id": "purpose.tourist_benefits_access.name"
    },
    {
        "id": "FAMILY_BENEFITS_ELIGIBILITY",
        "name_l10n_id": "purpose.family_benefits_eligibility.name"
    },
    {
        "id": "UNEMPLOYMENT_BENEFITS_ELIGIBILITY",
        "name_l10n_id": "purpose.unemployment_benefits_eligibility.name"
    },
    {
        "id": "ECONOMIC_SUPPORT_ALLOCATION",
        "name_l10n_id": "purpose.economic_support_allocation.name"
    },
    {
        "id": "DIGITAL_VOUCHER_USE",
        "name_l10n_id": "purpose.digital_voucher_use.name"
    },
    {
        "id": "PURCHASE_INCENTIVE_USE",
        "name_l10n_id": "purpose.purchase_incentive_use.name"
    },
    {
        "id": "CASHBACK_ELIGIBILITY_VERIFICATION",
        "name_l10n_id": "purpose.cashback_eligibility_verification.name"
    },
    {
        "id": "HEALTHCARE_BONUS_ACCESS",
        "name_l10n_id": "purpose.healthcare_bonus_access.name"
    },
    {
```

```
            "id": "MENTAL_HEALTH_VOUCHER_USE",
            "name_l10n_id": "purpose.mental_health_voucher_use.name"
        },
        {
            "id": "SPORTS_VOUCHER_USE",
            "name_l10n_id": "purpose.sports_voucher_use.name"
        },
        {
            "id": "ACCESS_PERMIT",
            "name_l10n_id": "purpose.access_permit.name"
        }
    ]
}
```

Nota

Per una gestione migliore e più efficiente della localizzazione della Tassonomia, un'Entità che la consulta DOVREBBE:

- Scaricare la versione base della Tassonomia (compatta, senza localizzazioni) utilizzando l'endpoint `.well-known/taxonomy`.
- Determinare la lingua preferita dall'Utente.
- Scaricare solo i bundle di localizzazione necessari.
- Unire dinamicamente il contenuto localizzato con la struttura della Tassonomia.

Di seguito è fornito un esempio non normativo dell'output di un bundle di localizzazione:

```
{
  "taxonomy.name": "IT-Wallet Taxonomy",
  "taxonomy.description": "Hierarchical classification system for Digital Credentials in the IT
-Wallet ecosystem",
  "domain.identity.name": "Identity",
  "domain.identity.description": "Credentials that establish or confirm a person's legal identi
ty and personal, civil or legal status.",
  "class.identification_documents.name": "Identification Documents",
  "purpose.identity_verification.name": "Identity verification",
  "domain.authentication.name": "Authentication",
  "domain.authentication.description": "Credentials that attest authorisation to access restric
ted physical or digital spaces, services or resources.",
  "class.access.name": "Access",
  "purpose.access_permit.name": "Access permit verification",
  "...": "..."
}
```

I bundle di localizzazione DEVONO essere disponibili all'URI composto aggiungendo il codice

locale e `.json` al valore `localization.base_uri` definito nella tassonomia. Ciascun bundle locale DEVE essere accessibile seguendo il pattern di denominazione **{locale_code}.json**, dove **{locale_code}** è sostituito con il codice locale corrispondente dall'array **available_locales**.

Un esempio non normativo dell'URI di localizzazione italiana per il bundle sarebbe **https://trust-registry.eid-wallet.example.it/.well-known/l10n/taxonomy/it.json**.

7.8 Registro degli Schema

Il **Registro degli Schemi** è l'inventario autorevole di tutti gli **Schemi di Credenziale** (JSON Schema per SD-JWT, CBOR Schema per mDOC) noti e accettati nell'ecosistema IT-Wallet. È gestito dal Trust Anchor e fornisce un'unica fonte verificabile per il recupero delle specifiche tecniche necessarie per il parsing, la validazione e la visualizzazione delle Credenziali Digitali.

Obiettivi del Registro degli Schemi:

1. **Centralizzazione degli Schemi**: Fornire un punto di accesso centralizzato per tutti gli schemi tecnici utilizzati dalle Credenziali Digitali.
2. **Integrità e Autenticità**: Garantire l'integrità e l'autenticità dei documenti di schema tramite digest crittografici.
3. **Interoperabilità**: Facilitare l'integrazione senza soluzione di continuità dei Fornitori di Wallet e delle Relying Party fornendo versioni di schema coerenti.
4. **Supporto al Ciclo di Vita della Credenziale**: Fungere da punto di riferimento verificabile per la validazione degli schemi durante l'emissione e la presentazione.

Struttura e Accesso al Registro degli Schemi:

Il Registro degli Schemi è accessibile tramite l'endpoint di discovery `.well-known/it-wallet-registry` nel campo *schema_registry*. Consente la discovery degli URI degli schemi e i relativi controlli di integrità crittografica.

Tabella 7.13: Campi di Primo Livello del Registro degli Schemi

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del Registro degli Schemi (es. `urn:schemas:it-wallet`).
version	OBBLIGATORIO. La versione del Registro degli Schemi (es. `1.0.0`).
last_modified	OBBLIGATORIO. Il timestamp che indica quando il Registro degli Schemi è stato aggiornato l'ultima volta (es. `2025-03-15T12:00:00Z`).

continues on next page

Tabella 7.13 – continua dalla pagina precedente

Nome Campo	Descrizione
schemas	OBBLIGATORIO. Un Array JSON in cui ogni voce è un Oggetto JSON che rappresenta una definizione di Schema di Credenziale. Ogni oggetto contiene i parametri definiti nella tabella "Parametri di Definizione dello Schema" di seguito, inclusi identificazione dello schema, specifiche del formato, URI e dati di verifica dell'integrità.

Tabella 7.14: Parametri di Definizione dello Schema

Nome Campo	Descrizione
id	OBBLIGATORIO. L'identificatore univoco dello schema (es. `mDL+mso_mdoc+org.iso.18013.5.1.mDL`).
version	OBBLIGATORIO. La versione della definizione dello schema (es. `1.0.0`).
credential_type	OBBLIGATORIO. L'identificatore univoco della tipologia di Credenziale Digitale (es. `mDL`, `pid`, `eid`).
format	OBBLIGATORIO. Il formato tecnico dello schema (es. `mso_mdoc`, `dc+sd-jwt`).
vct	CONDIZIONALE. È OBBLIGATORIO se il `format` è `dc+sd-jwt`, indicando il Tipo di Credenziale Verificabile (es. `urn:eudi:mDL:it:1`).
docType	CONDIZIONALE. È OBBLIGATORIO se il `format` è `mso_mdoc`, indicando il tipo di documento utilizzato (es. `org.iso.18013.5.1.mDL`).
schema_uri	OBBLIGATORIO. L'URI dove è possibile recuperare il documento dello schema (es. `https://trust-registry.it-wallet.example.it/.well-known/schemas/mdoc/mDL`).
schema_uri# integrity	OBBLIGATORIO. Digest crittografico del documento di schema per la verifica dell'integrità. Formato: `{digest_method}-{digest_value}` (es. `sha256-c8b708728e4c5756e35c03aeac257ca878d1f717d7b61f621be4d36dbd9b9c16`).
description	OPZIONALE. Una descrizione leggibile dello schema, che può essere localizzata (es. "Schema tecnico per la mobile Driving License in formato mdoc.").

Esempio di Registro degli Schemi:

Un esempio non normativo del payload del Registro degli Schemi:


```
{
    "id": "urn:schemas:it-wallet",
    "version": "1.0.0",
    "last_modified": "2025-03-15T12:00:00Z",
    "schemas": [
        {
            "id": "mDL+mso_mdoc+org.iso.18013.5.1.mDL",
            "version": "1.0.0",
            "credential_type": "mDL",
            "format": "mso_mdoc",
```

```
            "docType": "org.iso.18013.5.1.mDL",
            "schema_uri": "https://trust-registry.it-wallet.example.it/.well-known/schemas/mdoc
/mDL",
            "schema_uri#integrity": "sha256-c8b708728e4c5756e35c03aeac257ca878d1f717d7b61f621be
4d36dbd9b9c16",
            "description": "Schema tecnico per la mobile Driving License in formato mdoc."
        },
        {
            "id": "pid+dc+sd-jwt+urn:eudi:pid:it:1",
            "version": "1.0.0",
            "credential_type": "pid",
            "format": "dc+sd-jwt",
            "vct": "urn:eudi:pid:it:1",
            "schema_uri": "https://trust-registry.it-wallet.example.it/.well-known/schemas/sd-j
wt/pid",
            "schema_uri#integrity": "sha256-a1b2c3d4e5f67890...",
            "description": "Schema tecnico per la Person Identification Data (PID) in formato S
D-JWT."
        }
    ]
}
```


7.9 Integrazione del Registro e Riferimenti Incrociati

I componenti del registro sono interconnessi e operano congiuntamente per supportare il completo ecosistema delle Credenziali:

1. **Registro FA ↔ Tassonomia**: Le entità FA dichiarano le capacità utilizzando le classificazioni della tassonomia per una categorizzazione standardizzata.
2. **Registro FA ↔ Catalogo**: Le tipologie di Credenziale fanno riferimento alle capacità FA per la validazione della fonte dati.
3. **Catalogo ↔ Tassonomia**: Le voci del Catalogo specificano domini e finalità della tassonomia per la discovery e l'autorizzazione.
4. **Registro della Federazione ↔ Tutti i Componenti**: Fornisce la validazione crittografica della fiducia per tutte le operazioni di registro e l'autenticazione delle entità.
5. **Registro degli Schemi ↔ Emittenti/RP**: Fornisce il collegamento verificabile a tutte le specifiche di formato delle Credenziali note nell'ecosistema.

7.10 Percorsi di Utilizzo dell'Infrastruttura del Registro

I componenti dell'Infrastruttura del Registro sono progettati per supportare diverse fasi operative dell'ecosistema IT-Wallet, ciascuna con specifiche interazioni tra le entità. I principali Percorsi di seguito illustrano le interazioni con l'Infrastruttura del Registro.

7.10.1 Navigazione del Catalogo

Questo percorso di *Navigazione del Catalogo* supporta gli Utenti (sia utenti umani tramite un'**Istanza Wallet** che sistemi automatizzati come **Relying Party** o portali web) nella discovery e nella selezione delle Credenziali Digitali disponibili.

1. **Accesso all'Endpoint di Discovery**: L'entità (es. un Fornitore di Wallet o un portale informativo) accede all'*Endpoint di Discovery del Registro* (`.well-known/it-wallet-registry`) per ottenere l'URI del **Catalogo delle Credenziali Digitali** e della **Tassonomia**.
2. **Navigazione e Selezione**:

- **Discovery delle Credenziali**: L'entità naviga l'elenco delle Credenziali (campo `credentials`) per identificare le tipologie di Credenziale rilevanti (es. `pid`, `eid`, `mDL`) e, se necessario, utilizza le informazioni sulla **Tassonomia** per navigare la gerarchia e fornire diverse localizzazioni.
- **Metadati dell'Emittente**: L'entità estrae l'**Identificatore dell'Emittente** (*entity_id* nel campo *issuers*) associato alla Credenziale desiderata.
- **Consultazione dei Dettagli**: Per ottenere informazioni complete e requisiti tecnici specifici, l'entità accede alla **Configurazione dell'Entità** (Metadati dell'Emittente) utilizzando l'identificatore recuperato.

3. **Azione Finale**: L'entità può quindi utilizzare i metadati per mostrare le informazioni del catalogo a un Utente (o utilizzare le informazioni in altro modo).

7.10.2 Emissione di Credenziali

Questo percorso definisce come un Emittente di Credenziali utilizza l'Infrastruttura del Registro per preparare ed emettere una Credenziale Digitale conforme.

1. **Identificazione dei Requisiti**: L'EI consulta il **Catalogo delle Credenziali Digitali** per i requisiti tecnici della tipologia di Credenziale da emettere (es. `max_validity_days`, `min_loa`).
2. **Risoluzione di Schemi e Claims**:

- L'EI consulta il **Registro degli Schemi** per recuperare la specifica tecnica del formato e dello schema (es. JSON Schema per SD-JWT) richiesta dal Catalogo, verificandone la validità e l'integrità tramite l'hash (*schema_uri#integrity*).
- L'EI accede al **Registro dei Claims** per recuperare le definizioni semantiche standardizzate e i formati dei dati (tipi di dati) degli attributi (claim) necessari.

3. **Recupero dei Dati Autentici**:

- L'EI consulta il **Registro delle Fonti Autentiche (FA)** per identificare la **Fonte Autentica** (FA) autorizzata per il dataset richiesto. Il Registro FA fornisce l'`entity_id` della FA e i dettagli tecnici dell'interfaccia (*integration_endpoint*, *integration_method*).
- L'EI consulta la specifica dell'endpoint FA per implementare l'integrazione necessaria al recupero dei dati dell'Utente per popolare la Credenziale Digitale.

4. **Emissione della Credenziale**: L'EI utilizza i dati recuperati, gli schemi validati e i formati specificati per generare e firmare la Credenziale Digitale nel formato corretto (es. SD-JWT o mDOC).

7.10.3 Presentazione e Verifica delle Credenziali

Questo percorso descrive come un'**Istanza Wallet** e una **Relying Party (RP)** interagiscono con l'Infrastruttura del Registro quando una Credenziale Digitale deve essere presentata da un Utente.

1. **Autorizzazione e Selezione del Wallet**:

- Il Wallet riceve una Richiesta di Presentazione dalla RP, verifica la validità della richiesta confrontando i *claim* richiesti con le *Politiche di Autorizzazione* relative alla RP.
- Il Wallet consulta il **Catalogo delle Credenziali Digitali** e la **Tassonomia** per verificare i *Domini*, le *Classi* e le *Finalità* associate alle tipologie di Credenziale in suo possesso, valutando quali Credenziali sono adatte alla richiesta.
- Il Wallet verifica se gli attributi richiesti (claim) sono disponibili e autorizzati alla divulgazione in base alla politica della richiesta (scenari **Credential-Specific** o **Credential-Agnostic**).
- L'Utente autorizza il rilascio degli attributi selettivamente divulgati selezionati. Il Wallet quindi confeziona e presenta la Credenziale Digitale alla RP.

2. **Discovery e Integrità**:

- La RP riceve la Credenziale Digitale dall'Utente.
- La RP consulta il **Registro della Federazione** tramite l'endpoint del Trust Anchor (*federation_resolve*, *federation_trust_mark_status*) per verificare la **fiducia**

crittografica (Trust Mark) dell'Emittente e del Fornitore di Wallet come definito nella Sezione *L'Infrastruttura di Trust* (pagina 117).

- La RP consulta il **Registro degli Schemi** per scaricare lo schema della Credenziale presentata (*schema_uri*), verificandone l'integrità (*schema_uri#integrity*).

3. **Validazione dello Schema e della Politica Finale**:

- La RP utilizza lo schema recuperato per validare la struttura della Credenziale e i tipi di dati degli attributi rivelati.
- La RP esegue la verifica finale per assicurarsi che gli attributi presentati siano conformi ai requisiti specifici della richiesta iniziale e alla politica di autorizzazione.

4. **Accettazione o Rifiuto**: Sulla base della validazione crittografica, della conformità allo schema e dell'autorizzazione basata su politiche, la RP accetta o rifiuta la Credenziale per l'accesso al servizio.

CAPITOLO 8

Onboarding delle Entità

Questa sezione definisce le specifiche tecniche per la gestione del ciclo di vita delle entità nell'ecosistema IT-Wallet, basate sull'**Infrastruttura del Registro** definita in *Infrastruttura del Registro* (pagina 159). Sono incluse le procedure di onboarding iniziale, le operazioni di gestione continuativa, quali aggiornamenti e modifiche dei dati, e i processi di uscita dalla federazione.

Per una panoramica di alto livello del processo di onboarding, vedere *Sistema di Onboarding* (pagina 25). La sezione *Onboarding Journey Maps* (pagina 32) fornisce una mappa del percorso di onboarding dal punto di vista degli operatori dell'Entità.

8.1 Panoramica

8.1.1 Architettura del Sistema di Onboarding delle Entità

L'ecosistema IT-Wallet opera su un'infrastruttura di fiducia federata, che richiede alle entità partecipanti di stabilire una fiducia reciproca prima di intraprendere qualsiasi interazione che coinvolga attributi dell'Utente.

Il framework di onboarding definisce procedure di registrazione tecnica in base alla tipologia di partecipante:

1. Le Fonti Autentiche seguono processi di registrazione incentrati sui dati.
2. Le Entità Operative (Emittenti di Credenziali, Relying Party, Fornitori di Wallet) seguono procedure di instaurazione della fiducia.

8.1.2 Tipologie di Entità e Percorsi di Onboarding

La seguente tabella riepiloga le tipologie di entità, i loro ruoli e i relativi percorsi di onboarding:

Tabella 8.1: Tipologie di Entità e Percorsi di Onboarding

Tipologia di Entità	Ruolo Principale	Percorso di Onboarding	Requisiti Chiave
Fonti Autentiche	Fornitori di dati autorevoli per gli attributi delle Credenziali Digitali	*Procedura di registrazione delle Fonti Autentiche* (pagina 244)	Validazione della titolarità dei dati; integrazione API (PDND)
Emittenti di Credenziali	Generano ed emettono Credenziali Digitali utilizzando i dati delle Fonti Autentiche	*Processo di Onboarding delle Entità Federate* (pagina 256)	Conformità all'Infrastruttura di Fiducia IT-Wallet; vedere *L'Infrastruttura di Trust* (pagina 117)
Relying Party	Verificano le Credenziali Digitali per l'accesso ai servizi	*Processo di Onboarding delle Entità Federate* (pagina 256)	Conformità all'Infrastruttura di Fiducia IT-Wallet; vedere *L'Infrastruttura di Trust* (pagina 117)
Fornitori di Wallet	Forniscono soluzioni wallet ai cittadini	*Processo di Onboarding delle Entità Federate* (pagina 256)	Conformità all'Infrastruttura di Fiducia IT-Wallet (vedere *L'Infrastruttura di Trust* (pagina 117)); capacità di Attestazione del Wallet (vedere *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304))
Istanze di Wallet	Applicazioni di wallet digitale a livello utente	Registrazione indiretta tramite il Fornitore di Wallet, vedere *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304)	Attestazione del Wallet da un Fornitore di Wallet affidabile

8.1.3 Registrazione Amministrativa e Tecnica

Il processo di onboarding segue un approccio strutturato in più fasi:

1. **Registrazione Amministrativa**: Tutte le entità DEVONO completare la registrazione amministrativa iniziale che ne valida la posizione giuridica, la conformità normativa e l'idoneità organizzativa a partecipare all'ecosistema IT-Wallet.
2. **Registrazione Tecnica**: A seguito dell'approvazione amministrativa, le entità effettuano la registrazione tecnica attraverso percorsi specializzati:

- **Registrazione Fonte Autentica**: Procedure di registrazione incentrate sui dati con validazione dell'integrazione API.
- **Registrazione in Federazione**: Instaurazione della fiducia crittografica come definito nella Sezione *L'Infrastruttura di Trust* (pagina 117).

3. **Integrazione con il Registro IT-Wallet**:

- **Integrazione nel Registro dei Claims**: Le Fonti Autentiche selezionano definizioni standardizzate di claim dal Registro dei Claims durante la dichiarazione delle capacità.
- **Integrazione nella Tassonomia**: Tutte le entità utilizzano la classificazione gerarchica della Tassonomia (domini, classi, finalità) per strutturare organizzativamente le Credenziali.
- **Integrazione nel Registro delle Fonti Autentiche**: Le Fonti Autentiche vengono registrate con i claim e le capacità dichiarate, abilitando la discovery e il coordinamento degli Emittenti di Credenziali.
- **Integrazione nel Registro della Federazione**: Le entità operative vengono incluse per la validazione della fiducia durante le operazioni sulle Credenziali.
- **Integrazione nel Catalogo**: I tipi di Credenziale vengono pubblicati in *Struttura del Catalogo degli Attestati Elettronici* (pagina 206) in base alle politiche dell'Organismo di Vigilanza per l'idoneità alla discovery pubblica.

Tutti i componenti del registro e le loro interazioni sono descritti in dettaglio in *Infrastruttura del Registro* (pagina 159).

8.2 Processo di Registrazione delle Fonti Autentiche

Le Fonti Autentiche DEVONO completare un processo di registrazione strutturato per confermare il loro status di fornitori di dati autorevoli nell'ecosistema IT-Wallet. Tale processo include la specifica dei requisiti e la validazione procedurale, come descritto in *Journey dell'Operatore della Fonte Autentica* (pagina 34).

8.2.1 Requisiti di Registrazione delle Fonti Autentiche

Le Fonti Autentiche DEVONO rispettare i seguenti requisiti tecnici:

- **Conformità dei Claims**:
  - **Adozione del Registro dei Claims**: Le Entità DEVONO utilizzare identificatori standardizzati del Registro dei Claims nelle risposte ai dati. La mappatura personalizzata dei claim non è consentita.
- **Standard di Integrazione API**:
  - **Entità Pubbliche**: DEVONO integrarsi tramite la piattaforma PDND con un'implementazione dell'e-service conforme alle specifiche nazionali.
  - **Entità Private**: DEVONO fornire un documento di Specifica OAS3 completo che includa il framework di autorizzazione, gli schemi di richiesta/risposta, i meccanismi di gestione degli errori e un ambiente sandbox per i test.
- **Standardizzazione del Formato di Risposta**:
  - **Formato Standard dei Claims**: Le Entità DEVONO utilizzare gli identificatori e i formati del Registro dei Claims in tutte le risposte ai dati.
  - **Mappatura degli Stati**: Le Entità DEVONO gestire una mappatura chiara tra i propri stati interni e gli stati standard delle Credenziali (valida, sospesa, revocata).
- **Sicurezza e Garanzia della Qualità**:
  - **Standard di Sicurezza**: Le Entità DEVONO implementare TLS 1.3 o superiore con robusti meccanismi di autenticazione, perfect forward secrecy e algoritmi crittografici conformi agli standard di sicurezza attuali ed emergenti, con riservatezza e integrità end-to-end di tutte le comunicazioni, mantenendo la conformità ai requisiti normativi in evoluzione e alle best practice di settore.
  - **Evidenza di Autenticazione dell'Utente**: Le Entità POSSONO richiedere all'Emittente di Credenziali un'evidenza di autenticazione dell'Utente prima di concedere l'accesso agli e-service per l'ottenimento degli attributi dell'Utente.
  - **Qualità dei Dati**: Le Entità DEVONO specificare la frequenza di aggiornamento e fornire garanzie sull'aggiornamento dei dati.

– **Audit Trail**: Le Entità DEVONO implementare capacità di logging per tutte le attività di accesso e fornitura dei dati.

8.2.2 Requisiti sulle informazioni di registrazione delle Fonti Autentiche

Nota

Solo le Fonti Autentiche italiane possono essere registrate nella fase attuale di IT-Wallet.

Durante la registrazione, le Fonti Autentiche DEVONO fornire le seguenti informazioni:

Tabella 8.2: Requisiti Informativi per la Registrazione delle Fonti Autentiche

Categoria Informativa	Descrizione ed Esempi
Informazioni sull'Organizzazione	**OBBLIGATORIO**. Dettagli dell'organizzazione inclusi: • Nome dell'organizzazione, tipologia ("public" o "private") e paese (ISO 3166-1 alpha-2). • Codici identificativi amministrativi, come il codice di registrazione IPA (OBBLIGATORIO solo per le Fonti Autentiche pubbliche) e l'identificativo legale (codice fiscale o partita IVA). • Informazioni di contatto, compresi riferimenti tecnici e amministrativi, quali indirizzi email, URI della homepage, URI dell'informativa sulla privacy.
Dichiarazione delle Capacità sui Dati	**OBBLIGATORIO**. Claim disponibili: • Array di identificatori di claim del Registro dei Claims forniti dalla Fonte Autentica (es. `["given_name", "family_name", "driving_privileges"]`). • Finalità previste per la verifica, utilizzando gli identificatori di finalità della tassonomia (es. `["DRIVING_RIGHTS_VERIFICATION"]`).
Dettagli di Implementazione API	**OBBLIGATORIO**. Informazioni dettagliate sull'integrazione: • Framework di autorizzazione per l'accesso API. • Definizioni API quali Formati di Richiesta/Risposta, inclusa la gestione degli errori.
Capacità di Fornitura dei Dati	**OBBLIGATORIO**. Indica se la Fonte Autentica supporta la fornitura immediata/differita dei dati (booleano).

continues on next page

Tabella 8.2 – continua dalla pagina precedente

Categoria Informativa	Descrizione ed Esempi
Informazioni per l'Utente	**OPZIONALE**. Testo in formato Markdown contenente informazioni in linguaggio naturale sui vincoli o limitazioni nella disponibilità dei dati. Ad esempio, se il database della Fonte Autentica contiene solo dati registrati dopo una data specifica, questa informazione DEVE essere comunicata agli Utenti. **Esempio**: "I dati della patente di guida sono disponibili per le patenti rilasciate dopo il 1° gennaio 2020. Per le patenti più vecchie, contattare l'ufficio motorizzazione locale.".
Proprietà di Visualizzazione	**OPZIONALE**. Suggerimenti di branding visivo per le Credenziali Digitali che utilizzano i dati della Fonte Autentica: • Colore di sfondo per le Credenziali Digitali in formato esadecimale (es. `"#003d82"`). • URI del logo con verifica dell'integrità crittografica per il branding delle Credenziali Digitali. • URI del modello visivo con verifica dell'integrità per la presentazione delle Credenziali Digitali.

8.2.3 Procedura di registrazione delle Fonti Autentiche

La registrazione della Fonte Autentica segue un processo tecnico descritto di seguito.

Fase 1 - Predisposizione del Pacchetto di Registrazione: L'Entità predispone le informazioni di registrazione conformemente alla tabella dei requisiti sopra riportata. Di seguito è fornito un esempio non normativo delle informazioni in formato JSON.

Tabella 8.3: Campi di Primo Livello del Registro delle Fonti Autentiche

Nome Campo	Descrizione
id	OBBLIGATORIO. Identificatore univoco del Registro delle Fonti Autentiche (es. `urn:authentic-sources:it-wallet`).
version	OBBLIGATORIO. La versione del Registro delle Fonti Autentiche (es. `1.0.0`).
last_modified	OBBLIGATORIO. Il timestamp che indica quando l'elenco è stato aggiornato l'ultima volta (es. `2025-03-15T12:00:00Z`).

continues on next page

Tabella 8.3 – continua dalla pagina precedente

Nome Campo	Descrizione
localization	OBBLIGATORIO. Oggetto di configurazione della localizzazione contenente: • **default_locale**: Codice locale predefinito (es. `it`). • **available_locales**: Array dei codici locale supportati (es. `["en", "it"]`). • **base_uri**: URI base per il recupero del bundle di localizzazione (es. `https://trust-registry.eid-wallet.example.it/.well-known/l10n/authentic-sources/`). • **version**: Versione del formato del bundle di localizzazione.
authentic_sources	OBBLIGATORIO. Un Array JSON in cui ogni voce è un Oggetto JSON che rappresenta un'entità Fonte Autentica. Ogni oggetto contiene i parametri definiti nella tabella "Parametri delle Fonti Autentiche" di seguito, inclusi identificazione dell'entità, informazioni organizzative, capacità sui dati e metodi di integrazione.

Tabella 8.4: Parametri delle Fonti Autentiche

Parametro	Tipo	Descrizione
entity_id	string	OBBLIGATORIO. Identificatore univoco che segue lo schema normativo: `https://{organization_domain}[/{optional_path}]`.
organization_info	oggetto JSON	OBBLIGATORIO. Dettagli della persona giuridica e metadati organizzativi.
organization_info. organization_name	JSON Object Array	OBBLIGATORIO. Array di oggetti che contengono il nome dell'organizzazione in più lingue. DEVE contenere i claim `locale` e `name`.
organization_info. organization_type	string	OBBLIGATORIO. Classificazione dell'entità: `"public"` o `"private"`.
organization_info. ipa_code	string	OBBLIGATORIO solo per FA Pubbliche. Codice di registrazione IPA per gli enti pubblici.
organization_info. legal_identifier	string	OBBLIGATORIO. Identificativo di registrazione legale (Codice Fiscale/Partita IVA, o equivalente identificativo nazionale per entità straniere).
organization_info. homepage_uri	string	OBBLIGATORIO. URL della homepage dell'organizzazione.
organization_info. contacts	Array di stringhe	OBBLIGATORIO. Array di indirizzi email di contatto per almeno un referente di supporto utenti, uno applicativo e uno sistemistico.

continues on next page

Tabella 8.4 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
organization_info.dpa_contact	string	OBBLIGATORIO. Indirizzo email del DPA della Fonte Autentica.
organization_info.policy_uri	string	OBBLIGATORIO. URL del documento di informativa sulla privacy.
organization_info.tos_uri	string	OPZIONALE. URL del documento dei termini di servizio.
organization_info.organization_country	string	OBBLIGATORIO. Codice paese ISO 3166-1 alpha-2 a due lettere dell'organizzazione.
organization_info.logo_uri	string	OPZIONALE. URL dell'immagine del logo dell'organizzazione.
organization_info.logo_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo per la verifica dell'integrità. OBBLIGATORIO se `logo_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
organization_info.logo_alt_text	JSON Object Array	OPZIONALE. Array di oggetti contenente il testo alternativo per l'immagine del logo dell'organizzazione in più lingue. DEVE contenere i claim `locale` e `description`.
organization_info.logo_extended_uri	string	OPZIONALE. URL dell'immagine del logo esteso dell'organizzazione.
organization_info.logo_extended_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo esteso per la verifica dell'integrità. OBBLIGATORIO se `logo_extended_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
organization_info.logo_extended_alt_text	JSON Object Array	OPZIONALE. Array di oggetti contenente il testo alternativo per l'immagine del logo esteso dell'organizzazione in più lingue. DEVE contenere i claim `locale` e `description`.
data_capabilities	Array di oggetti JSON	OBBLIGATORIO. Array contenente le specifiche delle capacità sui dati.
data_capabilities[].dataset_id	string	OBBLIGATORIO. Il *Dataset_id* nell'ambito della Fonte Autentica, che PUÒ essere utilizzato come parametro di query per il servizio `GetAttributeClaims`.
data_capabilities[].data_origin	JSON Object Array	OBBLIGATORIO. Array di oggetti contenente il nome leggibile dell'origine o del dipartimento che fornisce i dati in più lingue. DEVE contenere i claim `locale` e `name`.
data_capabilities[].intended_purposes	Array di stringhe	OBBLIGATORIO. Finalità aziendali soddisfatte, utilizzando gli identificatori di finalità della tassonomia (es. `["IDENTITY_VERIFICATION", "DRIVING_RIGHTS_VERIFICATION"]`).
data_capabilities[].available_claims	Array di stringhe	OBBLIGATORIO. Claim disponibili da questa capacità dati.
data_capabilities[].available_claims.claim_name	string	OBBLIGATORIO. Contiene il nome del claim.

continues on next page

Tabella 8.4 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
data_capabilities[]. available_claims. order	number	OBBLIGATORIO. Definisce l'ordine in cui le informazioni vengono mostrate.
data_capabilities[]. available_claims. mandatory	boolean	OBBLIGATORIO. Definisce se un claim è sempre disponibile o meno.
data_capabilities[]. integration_method	string	OBBLIGATORIO. Framework di autorizzazione utilizzato per l'accesso ai dati. DEVE essere `"pdnd"`.
data_capabilities[]. integration_ endpoint	string	OPZIONALE. Punto di accesso al servizio (endpoint PDND).
data_capabilities[]. api_specification	string	OPZIONALE. URL del documento di specifica OAS3 per questa capacità dati.
data_capabilities[]. data_provision	oggetto JSON	OPZIONALE. Capacità di fornitura dei dati e specifiche temporali.
data_capabilities[]. data_provision. immediate_flow	boolean	OBBLIGATORIO. Indica se la Fonte Autentica supporta la fornitura immediata dei dati.
data_capabilities[]. data_provision. deferred_flow	boolean	OBBLIGATORIO. Indica se la Fonte Autentica supporta la fornitura differita dei dati.
data_capabilities[]. data_provision. max_response_ time_minutes	integer	CONDIZIONALE. Tempo massimo in minuti per la risposta della Fonte Autentica a una richiesta di fornitura dati differita. OBBLIGATORIO se `deferred_flow` è `true`.
data_capabilities[]. data_provision. notification_ methods	Array di stringhe	CONDIZIONALE. Array dei metodi di notifica supportati dalla Fonte Autentica per la fornitura dati differita, come `"push"`, `"poll"`. OBBLIGATORIO se `deferred_flow` è `true`.
data_capabilities[]. user_information	JSON object Array	OPZIONALE. Array di Oggetti contenenti informazioni per l'Utente in più lingue sulla disponibilità dei dati. Questa stringa DEVE essere fornita dalla Fonte Autentica al Trust Anchor durante l'onboarding. La formattazione Markdown può essere testo semplice o una combinazione di testo e link. Ad esempio, se il database della Fonte Autentica contiene solo dati registrati *dopo* una data specifica, questa informazione DEVE essere comunicata tramite questa chiave. DEVE contenere i claim `locale` e `description`.

continues on next page

Tabella 8.4 – continua dalla pagina precedente

Parametro	Tipo	Descrizione
data_capabilities[]. administrative_ expiration_user_ info	JSON object Array	OPZIONALE. Array di Oggetti contenenti informazioni per l'Utente in più lingue sulla scadenza amministrativa dei dati ed eventualmente le azioni raccomandate. Questa stringa DEVE essere fornita dalla Fonte Autentica al Trust Anchor durante l'onboarding. La formattazione Markdown può essere testo semplice o una combinazione di testo e link. DEVE contenere i claim `locale`, `title` e `description`.
data_capabilities[]. allowed_states	JSON object Array	OPZIONALE. Array di Oggetti contenente informazioni per l'Utente in più lingue sullo stato corrente dei dati forniti ed eventualmente le azioni raccomandate. I valori degli stati sono definiti nella sezione *Token Status Lists* (pagina 435). DEVE contenere i claim `locale`, `title`, `description` e `<Status-Type-Value>`.
data_capabilities[]. service_ documentation_ uri	string	OPZIONALE. URL che punta alla documentazione del servizio della Fonte Autentica.
data_capabilities[]. update_frequency	string	OPZIONALE. Indica la frequenza con cui la Fonte Autentica aggiorna i propri dati. Valori possibili: `"real_time"` (aggiornamenti quasi in tempo reale, tipicamente entro minuti), `"daily"`, `"weekly"`, `"monthly"`, `"on_demand"`.
data_capabilities[]. logo_uri	string	OPZIONALE. URL dell'immagine del logo relativa ai dati.
data_capabilities[]. logo_uri#integrity	string	CONDIZIONALE. Digest crittografico della risorsa immagine del logo per la verifica dell'integrità. OBBLIGATORIO se `logo_uri` è presente. Formato: `{digest_method}-{digest_value}` (es. `"sha-256-abc123..."`).
data_capabilities[]. background_color	string	OPZIONALE. Valore stringa del colore di sfondo da visualizzare insieme ai dati.
data_capabilities[]. contacts	Array di stringhe	OPZIONALE. Array di indirizzi email di contatto del servizio clienti.

```
{
  "data_capabilities": [
    {
      "api_specification": "https://docs.gov.example/transport/api-oas3.yaml",
      "available_claims": [
        {
          "claim_name": "given_name",
          "mandatory": true,
          "order": 1
        },
        {
          "claim_name": "family_name",
```

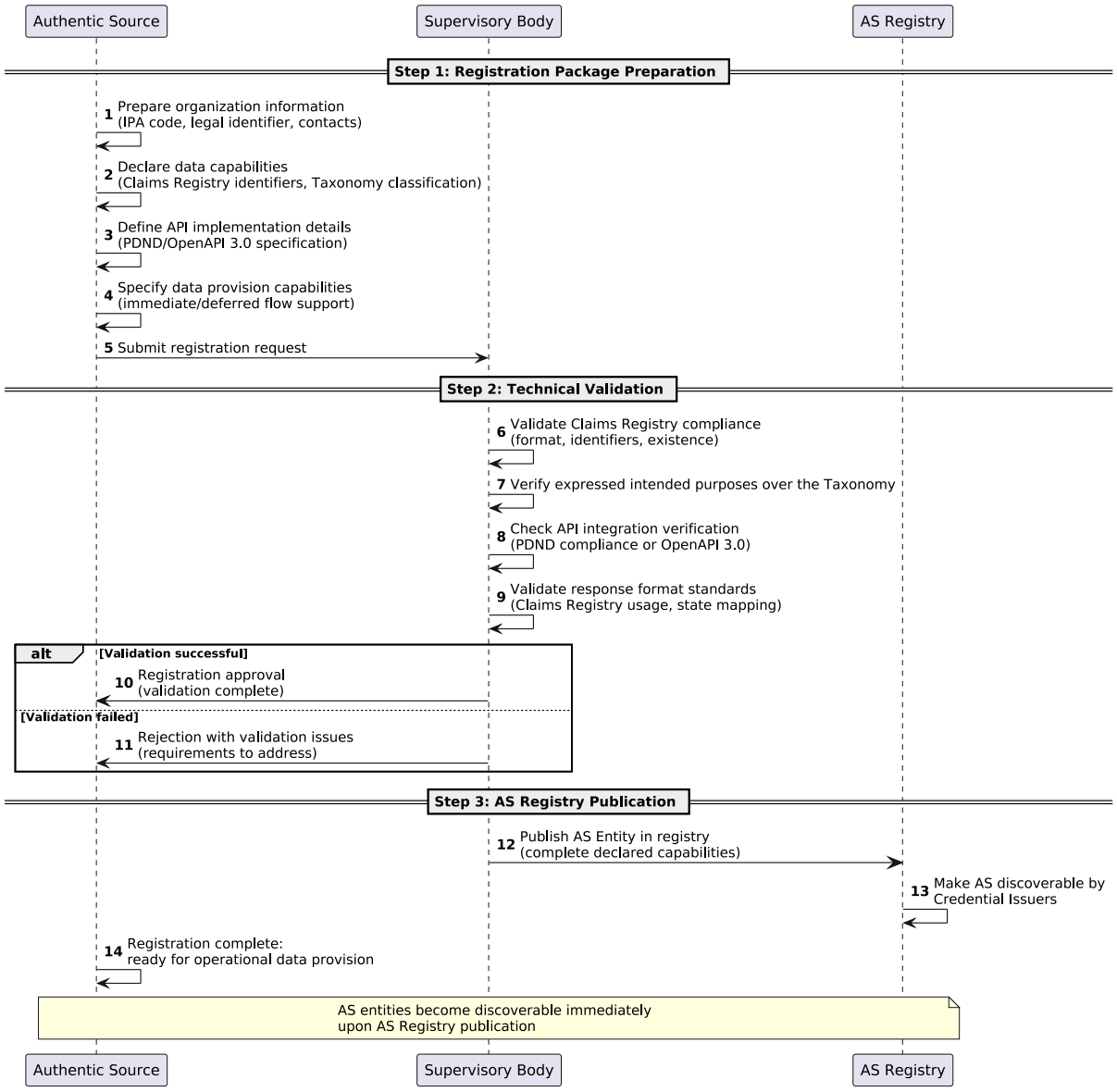


Fig. 8.1: Processo di Registrazione della Fonte Autentica.

```
    "mandatory": true,
    "order": 2
  },
  {
    "claim_name": "birth_date",
    "mandatory": true,
    "order": 3
  },
  {
    "claim_name": "birth_place",
    "mandatory": true,
    "order": 4
  },
  {
    "claim_name": "issue_date",
    "mandatory": true,
    "order": 5
  },
  {
    "claim_name": "document_iss_country",
    "mandatory": true,
    "order": 6
  },
  {
    "claim_name": "document_iss_authority",
    "mandatory": true,
    "order": 7
  },
  {
    "claim_name": "expiry_date",
    "mandatory": true,
    "order": 8
  },
  {
    "claim_name": "document_number",
    "mandatory": true,
    "order": 9
  },
  {
    "claim_name": "portrait",
    "mandatory": true,
    "order": 10
  },
  {
    "claim_name": "un_distinguishing_sign",
    "mandatory": true,
    "order": 11
```

```
        },
        {
          "claim_name": "restrictions_conditions",
          "mandatory": true,
          "order": 12
        },
        {
          "claim_name": "driving_privileges",
          "mandatory": true,
          "order": 13
        }
      ],
      "administrative_expiration_user_info": [
        {
        "locale": "it-IT",
        "title": "La tua Patente risulta scaduta",
        "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Rinnova il documento presso l'ente abilitato del tuo territori."
        },
        {
        "locale": "en-US",
        "title": "Your driving licence has expired",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Renew the document at the authorised office in your area."
        }
      ],
      "allowed_states": [
        {
        "locale": "it-IT",
        "0x00": "VALID",
        "title": "La tua Patente risulta valida",
        "description": "In questo momento puoi guidare le categorie di veicoli associate alla t
ua Patente"
        },
        {
        "locale": "en-US",
        "0x00": "VALID",
        "title": "Your driving licence is valid",
        "description": "At this time, you can drive the vehicle categories associated with your
licence."
        },
        {
        "locale": "it-IT",
        "0x01": "INVALID",
        "title": "La tua Patente risulta non valida",
        "description": "Rivolgiti all'ente preposto per maggiori dettagli."
        },
```

```
        {
        "locale": "en-US",
        "0x01": "INVALID",
        "title": "Your driving licence is not valid",
        "description": "Contact the competent authority for more details."
        },
        {
        "locale": "it-IT",
        "0x02": "SUSPENDED",
        "title": "La tua Patente risulta sospesa",
        "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Rivolgiti all'ente preposto per maggiori dettagli sul termine del periodo di so
spensione."
        },
        {
        "locale": "en-US",
        "0x02": "SUSPENDED",
        "title": "Your driving licence is suspended",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Contact the competent authority for more details on the end of the suspension peri
od."
        },
        {
        "locale": "it-IT",
        "0x0F": "ATTRIBUTE_UPDATE",
        "title": "La tua Patente risulta essere stata aggiornata",
        "description": "In questo momento non puoi guidare le categorie di veicoli associate al
la tua Patente. Aggiorna il documento su IT-Wallet."
        },
        {
        "locale": "en-US",
        "0x0F": "ATTRIBUTE_UPDATE",
        "title": "Your driving licence has been updated",
        "description": "At this time, you cannot drive the vehicle categories associated with y
our licence. Update the document in the IT-Wallet."
        }
      ],
      "background_color": "#12107c",
      "contacts": [
        "citizen-support@gov.example"
      ],
      "data_origin": [
        {
        "locale": "it-IT",
        "name":"Ufficio/Dipartimento XXX Autorità dei Trasporti"
        },
        {
```

```
        "locale": "en-US",
        "name":"Office/Department XXX Transport Authority"
        }
      ],
      "data_provision": {
        "deferred_flow": false,
        "immediate_flow": true
      },
      "dataset_id": "38832801",
      "integration_endpoint": "https://api.gov.example/transport/driving-license",
      "integration_method": "pdnd",
      "intended_purposes": [
        "IDENTITY_VERIFICATION",
        "PERSON_IDENTIFICATION",
        "DRIVING_RIGHTS_VERIFICATION",
        "LAW_ENFORCEMENT_CONTROLS"
      ],
      "logo_uri": "https://www.gov.example/assets/transport-data-logo.svg",
      "logo_uri#integrity": "sha-256-b1946ac92492d2347c6235b4d2611184e2a3f8b8c8f8f8e8f8f8f8f8f8
f8f8f8",
      "service_documentation_uri": "https://docs.gov.example/transport/api-docs",
      "update_frequency": "real_time",
      "user_information": [
        {
        "locale": "it-IT",
        "description": "Sono disponibili le patenti di guida rilasciate o rinnovate a partire d
al DD-MM-YYYY."
        },
        {
        "locale": "en-US",
        "description": "Only driving licences issued or renewed from DD-MM-YYYY will be availab
le."
        }
      ]
    }
  ],
  "entity_id": "https://transport-authority.gov.example",
  "organization_info": {
    "contacts": [
      "registry@transport-authority.gov.example",
      "technical-support@transport-authority.gov.example"
    ],
    "homepage_uri": "https://www.gov.example/transport",
    "ipa_code": "nta_001",
    "legal_identifier": "12345678901",
    "logo_extended_uri": "https://www.gov.example/assets/transport-authority-logo-extended.svg"
,
```

```
    "logo_extended_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998
e86f7f7a27ae3",
    "logo_uri": "https://www.gov.example/assets/transport-authority-logo.svg",
    "logo_uri#integrity": "sha-256-a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a2
7ae3",
    "organization_country": "XX",
    "organization_name": [
      {
      "locale": "it-IT",
      "name": "Autorità dei Trasporti"
      },
      {
      "locale": "en-US",
      "name": "Transport Authority"
      }
    ],

    "organization_type": "public",
    "policy_uri": "https://www.gov.example/transport/privacy-policy",
    "tos_uri": "https://www.gov.example/transport/tos"
  }
}
```


Fase 2 - Validazione Tecnica: L'Organismo di Vigilanza valida la registrazione inviata con focus su:

- **Conformità al Registro dei Claims**: Validazione del formato dei claim, degli identificatori e della loro esistenza nel Registro dei Claims.
- **Validazione della Tassonomia**: Verifica che le finalità dichiarate siano voci valide della tassonomia.
- **Verifica dell'Integrazione API**:
  - **Entità Pubbliche**: Verifica della conformità alla specifica PDND dell'e-service
  - **Entità Private**: Completezza della specifica OAS3 inclusi framework di autorizzazione, schemi richiesta/risposta, meccanismi di gestione errori e ambiente sandbox.
- **Standard del Formato di Risposta**: Verifica dell'utilizzo del formato del Registro dei Claims e delle specifiche di mappatura degli stati.

Fase 3 - Pubblicazione nel Registro delle Fonti Autentiche: Al termine della validazione positiva:

- L'Entità Fonte Autentica viene pubblicata nel Registro delle Fonti Autentiche con le capacità dichiarate complete.

- La Fonte Autentica diventa rilevabile dagli Emittenti di Credenziali per le richieste di integrazione.
- La Fonte Autentica è pronta per la fornitura operativa dei dati.

Nota

La registrazione della Fonte Autentica è completa e indipendente dall'integrazione con l'Emittente di Credenziali. Le Fonti Autentiche diventano rilevabili immediatamente dopo la pubblicazione nel Registro delle Fonti Autentiche, mentre la disponibilità delle Credenziali per gli Utenti finali dipende dall'autorizzazione amministrativa dalla Fonte Autentica all'Emittente di Credenziali, seguita dalla riuscita integrazione tecnica e dall'approvazione dell'Organismo di Vigilanza per l'idoneità al catalogo.

8.3 Processo di Autorizzazione dalla Fonte Autentica all'Emittente di Credenziali

A seguito dell'autorizzazione amministrativa dalla Fonte Autentica all'Emittente di Credenziali ottenuta durante la fase di registrazione amministrativa, le procedure di integrazione tecnica stabiliscono le connessioni API operative e i meccanismi di accesso ai dati tra gli Emittenti di Credenziali e le Fonti Autentiche.

L'integrazione tecnica comprende:

- **Configurazione degli Endpoint API**: Instaurazione di connessioni API sicure come specificato nelle specifiche tecniche delle Fonti Autentiche (e-service PDND).
- **Validazione della Mappatura dei Claims**: Verifica che l'implementazione dell'Emittente di Credenziali mappi correttamente le risposte della Fonte Autentica agli identificatori standardizzati del Registro dei Claims.
- **Test del Flusso dei Dati**: Validazione delle capacità di fornitura dei dati immediata o differita e dei meccanismi di gestione degli errori.
- **Implementazione della Sicurezza**: Configurazione dell'autenticazione, dell'autorizzazione e del logging di audit come richiesto dagli standard di sicurezza delle Fonti Autentiche.

8.4 Processo di Onboarding delle Entità Federate

Le Entità Federate, tra cui Emittenti di Credenziali, Relying Party e Fornitori di Wallet, devono completare le procedure di onboarding per diventare partecipanti idonei nell'ecosistema IT-Wallet. Tale processo stabilisce la fiducia distribuita mediante l'emissione di Certificati X.509, la validazione delle Catene di Fiducia e la verifica della conformità, come descritto in *L'Infrastruttura di Trust* (pagina 117).

8.4.1 Modello Gerarchico dell'Autorità di Federazione

La federazione IT-Wallet implementa un **modello di onboarding gerarchico** in cui le Entità Federate DEVONO essere registrate da uno dei seguenti attori:

1. **Trust Anchor**: L'autorità radice con la capacità di registrare direttamente qualsiasi Entità Federata.
2. **Intermediari**: Autorità delegate che registrano le Entità Foglia per conto del Trust Anchor.

Sia i Trust Anchor che gli Intermediari agiscono come **Autorità di Federazione** con le seguenti capacità di onboarding:

- **Emissione di Certificati X.509**: Emettono Certificati X.509 ai propri Subordinati Immediati con opportuni vincoli di denominazione come definito in *X.509 PKI* (pagina 147).
- **Applicazione di Politiche sui Metadati**: Applicano politiche sui metadati specifiche della federazione con **effetto a cascata** (le politiche del Trust Anchor prevalgono su quelle degli Intermediari).
- **Emissione di Trust Mark**: Emettono Trust Mark di Federazione attestanti le capacità e le concessioni dei Subordinati, conformemente al framework di fiducia.

Pertanto, le Entità Federate POSSONO essere registrate attraverso percorsi differenti:

- **Onboarding Diretto dal Trust Anchor**: L'Entità si registra direttamente presso il Trust Anchor.
- **Onboarding Mediato da un Intermediario**: L'Entità si registra presso un Intermediario appropriato.

8.4.2 Prerequisiti di Onboarding alla Federazione

Le Entità Federate DEVONO rispettare i seguenti requisiti tecnici prima di avviare il processo di onboarding:

- **Generazione delle Chiavi**: Le entità DEVONO generare almeno due coppie di chiavi mediante crittografia a curva ellittica come specificato in *Algoritmi Crittografici* (pagina 647):
  - **Coppia di Chiavi della Federazione**: Utilizzata per firmare le Configurazioni dell'Entità e attestare le chiavi specifiche dell'applicazione. Per le best practice di sicurezza e la continuità operativa, le entità DOVREBBERO mantenere più Chiavi di Entità della Federazione (almeno due) per abilitare la rotazione sicura delle chiavi e la risposta agli incidenti senza impatto sulle entità che hanno memorizzato nella cache le Configurazioni dell'Entità.
  - **Coppia/i di Chiavi Specifiche dell'Applicazione**: Utilizzate per le operazioni protocollari specifiche dell'entità, come l'emissione e la presentazione delle Credenziali.
- **Attestazione delle Chiavi Specifiche dell'Applicazione**: Le entità DEVONO creare Certificati X.509 auto-firmati per le proprie chiavi specifiche dell'applicazione utilizzando la Chiave Privata dell'Entità della Federazione.
- **Predisposizione della Configurazione dell'Entità**: Le entità DEVONO pubblicare una Configurazione dell'Entità (CE) firmata con la propria Chiave Privata dell'Entità della Federazione all'endpoint `/.well-known/openid-federation` come definito in *L'Infrastruttura di Trust* (pagina 117). La CE DEVE includere:
  - Un claim `jwks` contenente le Chiavi dell'Entità della Federazione in formato JSON Web Key (JWK).
  - Un claim `iss` con l'Identificatore dell'Entità della Federazione come definito in *Ruoli di Federazione* (pagina 118).
  - Un claim `sub` uguale al claim `iss`.
  - I claim `iat` ed `exp` che definiscono un intervallo di tempo valido.
  - Un claim `metadata` contenente i metadati specifici dell'entità organizzati per Tipologie di Metadati (vedere credential-issuer-entity-configuration:Entity Configuration del Fornitore di Attestati Elettronici, *Entity Configuration Relying Party* (pagina 372), o *Entity Configuration del Fornitore di Wallet* (pagina 326)) con chiavi specifiche dell'applicazione incluse nei campi `jwks` dei metadati e Certificati X.509 auto-firmati nei relativi claim `x5c`.
- **Richiesta di Firma del Certificato X.509 (CSR)**: Le entità DEVONO predisporre una Richiesta di Firma del Certificato X.509 (CSR) in formato PKCS #10 contenente **la**

Chiave dell'Entità della Federazione per l'emissione del Certificato X.509 da parte dell'Autorità di Federazione, come definito in *Emissione di Certificati X.509* (pagina 149).

8.4.3 Requisiti di Sicurezza per la Gestione delle Chiavi

Tutte le entità federate DOVREBBERO mantenere almeno due chiavi di firma attestate dal Trust Anchor:

- **Chiave Attiva**: Utilizzata per le operazioni di firma correnti
- **Chiave di Backup**: Disponibile per l'attivazione immediata durante incidenti o rotazione pianificata delle chiavi

Questo approccio a doppia chiave consente: - Rotazione sicura delle chiavi senza interruzione del servizio - Risposta rapida agli incidenti in caso di compromissione della chiave primaria - Continuità per le entità con Configurazioni dell'Entità in cache - Prevenzione di problemi di validazione durante le transizioni di chiave

La chiave di backup DEVE essere: - Registrata dal Trust Anchor prima del deployment - Pubblicata nel JWKS dell'entità accanto alla chiave attiva - Pronta per l'attivazione immediata senza ulteriori fasi di certificazione - Mantenuta con gli stessi standard di sicurezza della chiave attiva

8.4.4 Procedura di Onboarding alla Federazione

La procedura di onboarding alla federazione segue una procedura strutturata in 4 fasi e può essere eseguita dal Trust Anchor o da un Intermediario.

Nota

La seguente procedura si applica ai Fornitori di Wallet, agli Emittenti di Credenziali e alle Relying Party che desiderano effettuare l'onboarding nella federazione IT-Wallet. L'**Autorità di Federazione** si riferisce al Trust Anchor o all'Intermediario in base alle caratteristiche organizzative e alle politiche di governance della federazione.

Nota

Questa sezione copre i requisiti di registrazione tecnica. Tutte le informazioni amministrative, quali la validazione della persona giuridica, la conformità normativa e l'idoneità organizzativa, si considerano già raccolte e validate dall'Organismo di Vigilanza durante la fase di registrazione amministrativa, che esula dalla presente specifica tecnica. Esempi di informazioni

amministrative includono: denominazione legale, numeri di registrazione dell'impresa, persone di contatto, documentazione di conformità legale e autorizzazioni operative.

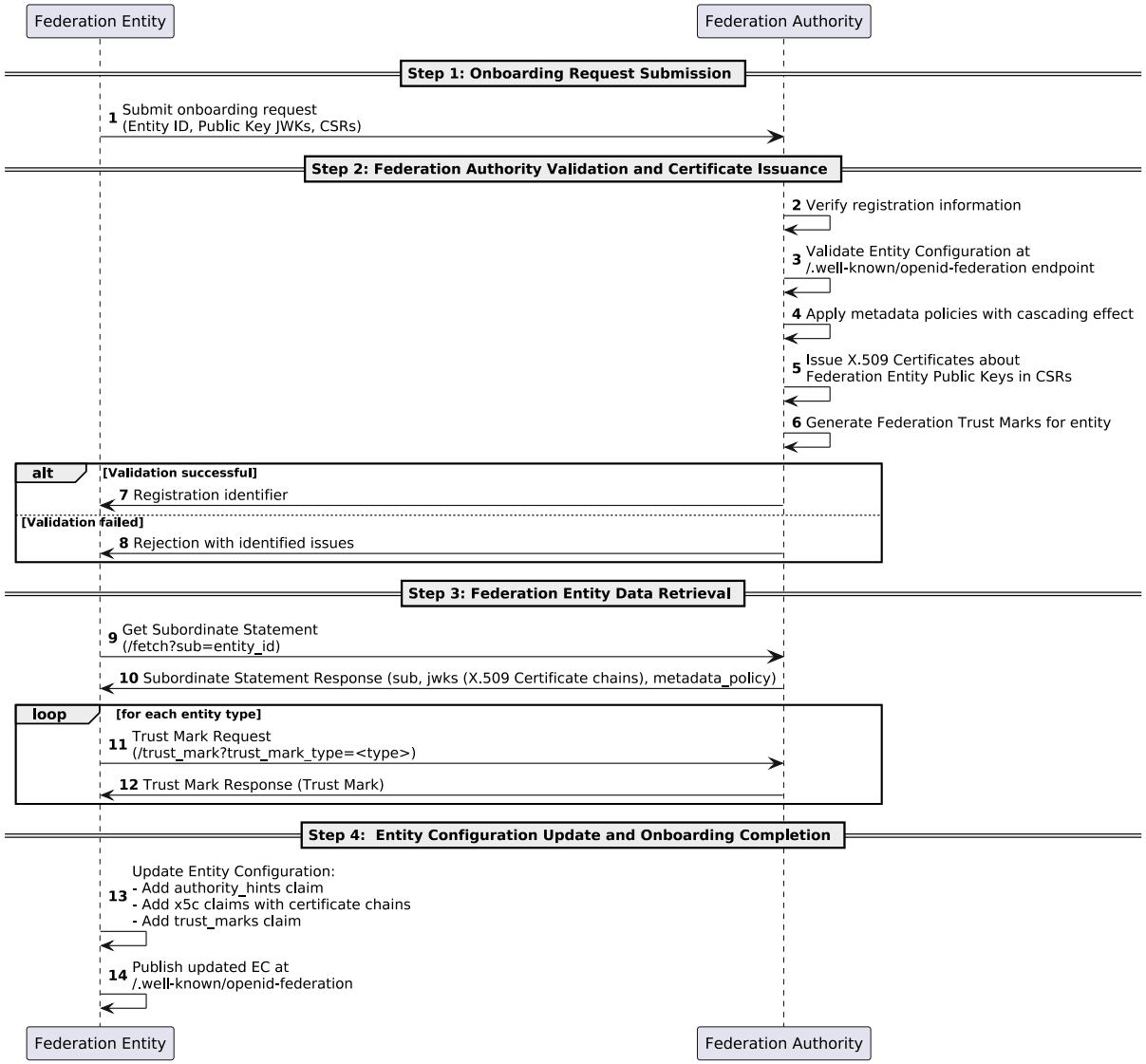


Fig. 8.2: Processo di Onboarding dell'Entità Federata.

Fase 1 - Invio della Richiesta di Onboarding: L'Entità Federata avvia il processo di onboarding inviando una richiesta di registrazione tecnica contenente le seguenti informazioni.

Tabella 8.5: Informazioni della Richiesta Tecnica di Onboarding alla Federazione

Categoria Informativa Tecnica	Requisiti e Descrizione
Identificatore dell'Entità Federata	**OBBLIGATORIO**. Un URL univoco che identifica l'Entità Federata come definito in *Ruoli di Federazione* (pagina 118).
Chiave dell'Entità Federata (JWK)	**OBBLIGATORIO**. Chiave pubblica a curva ellittica in formato JSON Web Key utilizzata per firmare le Configurazioni dell'Entità e attestare le chiavi specifiche dell'applicazione, con algoritmi crittografici specificati in *Algoritmi Crittografici* (pagina 647).
Richiesta di Firma del Certificato X.509 (CSR)	**OBBLIGATORIO**. CSR in formato PKCS #10 per l'emissione del Certificato X.509 da parte dell'Autorità di Federazione. La CSR DEVE: • Contenere **solo le Chiavi dell'Entità Federata**. • Essere firmata con la corrispondente Chiave Privata dell'Entità Federata. • Definire il Soggetto del Certificato X.509 con gli attributi richiesti come specificato in *Emissione di Certificati X.509* (pagina 149) per le Entità Federate.

Avvertimento

Prima di inviare la richiesta tecnica di onboarding, le Entità Federate DEVONO assicurarsi che il proprio endpoint `/.well-known/openid-federation` pubblichi una Configurazione dell'Entità valida (come definito in *Entity Configuration* (pagina 126)) firmata con la propria Chiave Privata dell'Entità Federata corrispondente alla Chiave dell'Entità Federata fornita nella richiesta. La Configurazione dell'Entità DEVE già includere le chiavi specifiche dell'applicazione nei metadati con Certificati X.509 auto-firmati nei claim `x5c`.

Un esempio non normativo della struttura delle informazioni tecniche che le Entità Federate inviano durante la richiesta di onboarding della Fase 1:


```
{
    "entity_id": "https://credential-issuer.example.gov",
    "entity_type": "credential_issuer",
    "jwks": {
        "keys": [
            {
                "kid": "k1d2e3f4g5h6i7j8",
                "kty": "EC",
                "crv": "P-256",
                "x": "8w1QwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw",
```

```
                "y": "QwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw"
            }
        ]
    },
    "certificate_signing_requests": {
        "k1d2e3f4g5h6i7j8": "-----BEGIN CERTIFICATE REQUEST-----\nMIIBWTCB/QIBADCBkDELMAkGA1UEB
hMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYD\nVQQHDARSb21hMRYwFAYDVQQKDA1FeGFtcGxlIE9yZzEkMCIGA1UEAwwbY3
JlZGVu\ndGlhbC1pc3N1ZXIuZXhhbXBsZS5nb3YwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNC\nAAR8w1QwQwQwQwQwQwQwQ
wQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\nQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw
QwQwQwQwQwQwQw\nQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\nQwQwQwQwQwQwQ
wQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQwQw\n-----END CERTIFICATE REQUEST-----",
    },
    "submission_timestamp": "2025-09-25T14:30:00Z"
}
```


L'esempio seguente mostra il contenuto decodificato della CSR:

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: CN=credentials.example.gov, OU=Digital Credentials, O=Example Organization,
        L=Roma, ST=Lazio, C=IT, emailAddress=technical@credentials.example.gov
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                ASN1 OID: prime256v1
                NIST CURVE: P-256
    Signature Algorithm: ecdsa-with-SHA256
```

Nota

Gli attributi del Soggetto della CSR DEVONO rispettare i requisiti specificati in *Emissione di Certificati X.509* (pagina 149) per le Entità Federate.

Nota

Le Chiavi dell'Entità Federata contenute nell'oggetto `jwks` e le chiavi pubbliche contenute nell'oggetto `certificate_signing_requests` DEVONO corrispondere. Le chiavi sono fornite in due formati: formato JWK e formato PKCS #10 CSR per l'emissione del Certificato X.509. Le chiavi specifiche dell'applicazione sono incluse solo nei metadati della Configurazione dell'Entità e NON DEVONO essere incluse nella richiesta di onboarding.

Nota

L'Endpoint della Configurazione dell'Entità è costruito automaticamente aggiungendo `/.well-known/openid-federation` all'Identificatore dell'Entità Federata (`entity_id`). Le Entità Federate non devono specificare separatamente questo endpoint nella richiesta di registrazione.

Fase 2 - Validazione dell'Autorità di Federazione ed Emissione del Certificato X.509: A seguito dell'invio della richiesta di onboarding, l'**Autorità di Federazione** DEVE eseguire:

- Verifica delle informazioni fornite nella richiesta di registrazione.
- Validazione della Configurazione dell'Entità, e dei metadati in essa contenuti, pubblicata all'endpoint `/.well-known/openid-federation` dell'entità (come definito in *L'Infrastruttura di Trust* (pagina 117)).
- **Applicazione delle Politiche sui Metadati**: Applicazione delle politiche sui metadati specifiche della federazione ai metadati dell'entità in base alle caratteristiche organizzative e all'ambito di autorizzazione come definito in *Subordinate Statement* (pagina 134). Quando registrata tramite un Intermediario, si applicano le politiche sia dell'Intermediario che del Trust Anchor, con le politiche del Trust Anchor prevalenti in caso di conflitti.
- **Emissione del Certificato X.509**: Certificazione della Chiave dell'Entità Federata con l'emissione del Certificato X.509 utilizzando l'infrastruttura di fiducia dettagliata in *Emissione di Certificati X.509* (pagina 149). Gli Intermediari DEVONO emettere Certificati X.509 utilizzando **vincoli di denominazione** appropriati che limitino l'uso di nomi DNS e URI ai soli propri subordinati.

In caso di esito positivo, l'entità riceve un identificativo di registrazione.

Nota

Se il processo di emissione fallisce, l'entità richiedente riceve una risposta con i problemi identificati da risolvere prima di inviare una nuova richiesta di onboarding.

Nota

Ciascuna entità può raccogliere la propria Catena di Certificati X.509 ottenendo il Certificato X.509 auto-firmato del Trust Anchor, incluso nell'oggetto `jwks` della Configurazione dell'Entità, e aggregando tutti i valori `x5c` pubblicati dalle entità superiori, dal Trust Anchor fino al suo superiore immediato. Questo consente all'entità di ricostruire la catena

di certificati completa necessaria per la validazione e l'instaurazione della fiducia all'interno della federazione.

Fase 3 - Recupero della Dichiarazione del Subordinato: come conferma del successo del processo di onboarding, l'entità DEVE recuperare la Dichiarazione del Subordinato che la riguarda dalla Federazione Immediata utilizzando l'endpoint `/fetch` come definito in *Endpoint API di Federazione* (pagina 123).

In questo modo, l'entità riceve la Catena di Certificati X.509 all'interno del claim `jwks` in cui:

- Il primo elemento è il Certificato X.509 contenente la Chiave dell'Entità Federata (emesso dall'Autorità di Federazione).
- **Per l'onboarding da Trust Anchor**: Il secondo elemento è il Certificato X.509 auto-firmato del Trust Anchor per la validazione del primo Certificato X.509.
- **Per l'onboarding da Intermediario**: Gli elementi aggiuntivi includono il Certificato X.509 dell'Intermediario e il Certificato auto-firmato del Trust Anchor, formando una catena di Certificati X.509 completa.
- Tutti i Certificati X.509 sono espressi in formato DER e codificati in Base64.

Esempio di risposta con catena di Certificati X.509 all'interno del claim `jwks`:


```
{
   "keys": [
     {
       "crv": "P-256",
       "kty": "EC",
       "x": "qrJrj3Af_B57sbOIRrcBM7br7wOc8ynj7lHFPTeffUk",
       "y": "1H0cWDyGgvU8w-kPKU_xycOCUNT2o0bwslIQtnPU6iM",
       "kid": "5t5YYpBhN-EgIEEI5iUzr6r0MR02LnVQ0OmekmNKcjY",
                   "x5c": ["MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQsw...", "MIIDQzCC
AuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQsw..."]
     }
   ]
 }
```


Esempio di richiesta fetch:

```
GET /fetch?sub=https%3A%2F%2Fcredentials.example.gov HTTP/1.1
Host: trust-anchor.example.gov
```

Nota

Se l'endpoint `/fetch` risponde con codice di stato `400` o `404`, l'entità DEVE risolvere i problemi descritti nel messaggio di risposta prima di richiamare nuovamente l'endpoint fetch.

Inoltre, l'entità può recuperare i Trust Mark tramite l'endpoint Trust Mark come definito in *Endpoint API di Federazione* (pagina 123).

Fase 4: Aggiornamento della Configurazione dell'Entità e Completamento dell'Onboarding

A seguito della richiesta fetch, l'**Autorità di Federazione** restituisce la Dichiarazione del Subordinato dell'entità, un JWT firmato contenente:

- **Metadati dell'Entità**: I metadati validati dell'entità con le politiche di federazione applicate.
- **Informazioni sulla Fiducia**: Informazioni sul rapporto di fiducia tra l'Autorità di Federazione e l'entità, inclusa la *x5c* emessa nell'oggetto *jwk* relativo al Subordinato.
- **Trust Mark**: I Trust Mark emessi dall'Autorità di Federazione per l'ambito operativo e le capacità specifiche dell'entità.

  Esempio di JWK di Federazione relativo al Subordinato incluso il Certificato X.509 emesso:

```
{
  "kid": "NsXymfIILEPR5Y0t",
  "kty": "EC",
  "x": "gXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l_g",
  "y": "l-6dcrIrFVdrzoY9cRJv9zNuFOR3MsDz6TSDhB0xEo4",
  "crv": "P-256",
  "x5c": [
    "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQsw..."
  ]
}
```

L'Entità Federata DEVE completare il processo di onboarding:

1. **Aggiornare e Pubblicare la Configurazione dell'Entità**:

   Il contenuto dell'endpoint `/.well-known/openid-federation` deve essere aggiornato per includere i seguenti claim:

   - Aggiungere un claim `authority_hints` con un Array JSON contenente l'Identificatore dell'Entità Federata dell'**Autorità di Federazione immediata** (Trust Anchor per l'onboarding diretto, o Intermediario per l'onboarding mediato) come definito in *Ruoli di Federazione* (pagina 118).
   - **Integrare i Trust Mark**: Includere i Trust Mark della Dichiarazione del Subordinato nella Configurazione dell'Entità usando il claim `trust_marks` come specificato in *Entity Configuration Foglie e Intermediari* (pagina 130).

Esempio di aggiunta di authority_hints:

```
{
  "iat": 1718207217,
  "exp": 1749743216,
  "iss": "https://credentials.example.gov",
  "sub": "https://credentials.example.gov",
  "authority_hints": ["https://trust-anchor.example.gov"],
  //...
}
```

Esempio di integrazione dei Trust Mark:

```
"trust_marks": [
  {
    "trust_mark_type": "https://trust-anchor.example.gov/trust_marks/federation-entity/credential-issuer",
    "trust_mark": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3RydXN0LWFuY2hvci5leGFtcGxlLmdvdiIsInN1YiI6Imh0dHBzOi8vY3JlZGVudGlhbC1pc3N1ZXIuZXhhbXBsZS5nb3YiLCJ0cnVzdF9tYXJrX3R5cGUiOiJodHRwczovL3RydXN0LWFuY2hvci5leGFtcGxlLmdvdi90cnVzdF9tYXJrcy9mZWRlcmF0aW9uLWVudGl0eS9jcmVkZW50aWFsLWlzc3VlciIsImlhdCI6MTcwMDAwMDAwMCwiZXhwIjoxNzMwMDAwMDAwfQ.abc123signature"
  }
]
```

Nota

Le Entità Federate possono ottenere Trust Mark in qualsiasi momento utilizzando il Federation Trust Mark Endpoint come definito in *Endpoint API di Federazione* (pagina 123).

Al completamento positivo della Fase 4, **l'onboarding dell'entità è completato con successo**. L'entità è ora operativa all'interno della federazione IT-Wallet e pronta per le attività operative.

Nota

Integrazione nel Registro della Federazione: Al completamento positivo dell'onboarding, l'Identificatore dell'Entità Federata dell'entità diventa rilevabile tramite i meccanismi di listing del Trust Anchor (come definito in *L'Infrastruttura di Trust* (pagina 117)), indicando la partecipazione attiva alla federazione. L'entità diventa parte dell'infrastruttura di federazione dettagliata in *Infrastruttura del Registro* (pagina 159).

8.4.5 Trust Mark della Federazione IT-Wallet

Le Entità Federate ricevono i Trust Mark della Federazione IT-Wallet al completamento positivo dell'onboarding. **I Trust Mark sono emessi dall'Autorità di Federazione** (Trust Anchor per l'onboarding diretto, Intermediario per l'onboarding mediato) attraverso il Federation Trust Mark Endpoint e fungono da attestazioni verificabili sulla conformità ai profili tecnici IT-Wallet e/o alle politiche di autorizzazione.

Tipologie e Schema dei Trust Mark

Le Entità POSSONO ricevere più Trust Mark per finalità diverse. Gli identificatori dei Trust Mark DEVONO seguire uno schema gerarchico che rispecchia l'ambito di autorizzazione:

`https://<federation_authority_domain>/trust_marks/<purpose>/<entity_type>`

Dove:

- `<federation_authority_domain>`: Il dominio dell'Autorità di Federazione emittente.
- `<purpose>`: La finalità del Trust Mark. La finalità `federation-entity` è **OBBLIGATORIA** per tutte le entità. Finalità aggiuntive dei Trust Mark POSSONO essere supportate, come `authorization_policy` per definizioni granulari dell'ambito operativo.
- `<entity_type>`: La tipologia dell'entità destinataria (es. `credential-issuer`, `relying-party`, `wallet-provider`).

Struttura dei Trust Mark

I Trust Mark nella Configurazione dell'Entità DEVONO essere rappresentati come oggetti JSON contenenti i seguenti claim:

Tabella 8.6: Claim dell'Oggetto Trust Mark (nella Configurazione dell'Entità)

Claim	Descrizione
trust_mark_type	**OBBLIGATORIO**. Identificatore della tipologia di Trust Mark secondo lo schema: `https://<federation_authority_domain>/trust_marks/<purpose>/<entity_type>`.
trust_mark	**OBBLIGATORIO**. Un JSON Web Token firmato che rappresenta il Trust Mark emesso dall'Autorità di Federazione.

Il JWT del Trust Mark (contenuto nel claim `trust_mark` di cui sopra) include i seguenti claim:

Tabella 8.7: Claim del JWT del Trust Mark

Claim	Descrizione
iss	**OBBLIGATORIO**. Autorità di Federazione emittente il Trust Mark (superiore immediato: Trust Anchor o Intermediario).
sub	**OBBLIGATORIO**. Identificatore dell'Entità Federata del soggetto.
trust_mark_type	**OBBLIGATORIO**. Identificatore univoco del Trust Mark, DEVE corrispondere al claim `trust_mark_type`.
iat	**OBBLIGATORIO**. Timestamp di emissione del Trust Mark.
exp	**OBBLIGATORIO**. Timestamp di scadenza del Trust Mark.
organization_type	**OBBLIGATORIO**. Tipologia organizzativa dell'entità (`public` o `private`).
vat_number	**RACCOMANDATO**. Partita IVA dell'entità (tipicamente per le organizzazioni private).
legal_identifier	**RACCOMANDATO**. Numero o identificativo di registrazione legale dell'entità (es. numero di registrazione dell'impresa, codice fiscale).
ipa_code	**RACCOMANDATO**. Codice IPA (Indice delle Pubbliche Amministrazioni) per le entità del settore pubblico.
organization_name	**RACCOMANDATO**. Denominazione completa dell'Entità Organizzativa.
email	**RACCOMANDATO**. Email istituzionale o PEC dell'organizzazione.
logo_uri	**OBBLIGATORIO**. URL che punta al *Trust Mark* (pagina 18) per scopi UI/UX.
ref	**OPZIONALE**. URL con informazioni web aggiuntive sul Trust Mark.

I seguenti esempi non normativi illustrano diversi contenuti del JWT del Trust Mark per l'appartenenza alla federazione e diverse politiche di autorizzazione:


```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://ci.public-authority.gov.example",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/
credential-issuer",
  "iat": 1718207217,
  "exp": 1749743216,
  "organization_type": "public",
  "ipa_code": "pub_001",
  "legal_identifier": "12345678901",
  "organization_name": "Public Authority Services",
  "email": "registry@public-authority.gov.example"
}
```

```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://rental.cars.example.com",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/authorization_poli
cy/relying-party",
  "iat": 1718207217,
```

```
  "exp": 1749743216,
  "organization_type": "private",
  "vat_number": "IT12345678901",
  "legal_identifier": "12345678901",
  "organization_name": "Premium Car Rental Services Ltd",
  "email": "compliance@rental.cars.example.com",
  "authorized_claims": ["given_name", "family_name", "driving_privileges"],
  "authorized_credential_types": ["mobile-driving-license"],
  "scope_restrictions": {
    "domains": ["MOBILITY_TRAVEL"],
    "purposes": ["DRIVING_RIGHTS_VERIFICATION"]
  }
}
```

```
{
  "iss": "https://trust-anchor.eid-wallet.example.it",
  "sub": "https://private-badge.ci.example.com",
  "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/authorization_poli
cy/credential-issuer",
  "iat": 1718207217,
  "exp": 1749743216,
  "organization_type": "private",
  "vat_number": "IT98765432101",
  "legal_identifier": "98765432101",
  "organization_name": "Badge Services Ltd",
  "email": "compliance@rprivate-badge.ci.example.com",
  "authorized_claims": ["given_name", "family_name", "company_id"],
  "authorized_credential_types": ["example-company-badge"],
  "scope_restrictions": {
    "domains": ["EMPLOYMENT"],
    "purposes": ["ACCESS_PERMIT"]
  }
}
```


Le Entità Federate DEVONO integrare i Trust Mark nella propria Configurazione dell'Entità utilizzando il claim `trust_marks` come specificato in *Entity Configuration Foglie e Intermediari* (pagina 130). Le Entità POSSONO ricevere più Trust Mark per diversi ambiti di autorizzazione.


```
{
  "iss": "https://credentials.example.gov",
  "sub": "https://credentials.example.gov",
  "jwks": {
   // contenuto jwks
  },
  "authority_hints": ["https://trust-anchor.eid-wallet.example.it"],
  "trust_marks": [
    {
```

```
      "trust_mark_type": "https://trust-anchor.eid-wallet.example.it/trust_marks/federation-entity/credential-issuer",
      "trust_mark": "eyJhbGciOiJFUzI1NiIsImtpZCI6IlRydXN0QW5jaG9yS2V5SWQiLCJ0eXAiOiJKV1QifQ..."
    }
  ],
  "metadata": {
   // contenuto dei Metadati
  }
}
```

```
{
  "iss": "https://healthcare-ci.example.gov",
  "sub": "https://healthcare-ci.example.gov",
  "jwks": {
   // contenuto jwks
  },
  "authority_hints": ["https://healthcare.intermediate.eid-wallet.example.it"],
  "trust_marks": [
    {
      "trust_mark_type": "https://healthcare.intermediate.eid-wallet.example.it/trust_marks/federation-entity/credential-issuer",
      "trust_mark": "eyJhbGciOiJFUzI1NiIsImtpZCI6IkhlYWx0aGNhcmVJbnRlcm1lZGlhdGVLZXlJZCIsInR5cCI6IkpXVCJ9..."
    }
  ],
  "metadata": {
   // contenuto dei Metadati
 }
}
```

Validazione dei Trust Mark

I partecipanti alla federazione validano lo stato dei Trust Mark tramite due meccanismi:

1. **Validazione Statica**: Verifica crittografica utilizzando la chiave pubblica dell'Autorità di Federazione emittente dalla Catena di Fiducia.
2. **Validazione Dinamica**: Verifica dello stato in tempo reale, contro eventuali revoche, utilizzando l'endpoint `/trust_mark_status` dell'Autorità di Federazione come definito in *Endpoint API di Federazione* (pagina 123).

Per le procedure complete di gestione dei Certificati X.509, inclusa la validazione della catena, la verifica delle revoche e la gestione del ciclo di vita, vedere *Operazioni di Gestione dei Certificati X.509* (pagina 271).

CAPITOLO 9

Operazioni di Gestione dei Certificati X.509

Questa sezione definisce le procedure operative per la gestione dei Certificati X.509 all'interno della federazione IT-Wallet, coprendo l'analisi delle catene di Certificati X.509, le procedure di validazione e la verifica di revoca. Queste procedure completano i processi di onboarding della federazione e supportano la gestione continua del ciclo di vita dei Certificati X.509 per tutti i partecipanti.

Per l'infrastruttura PKI X.509 fondamentale e le procedure di emissione dei certificati, vedere *X.509 PKI* (pagina 147).

9.1 Architettura PKI della Federazione

La federazione IT-Wallet opera con un'infrastruttura a chiave pubblica gerarchica dove:

- **Trust Anchor**: Agisce come Autorità di Certificazione X.509 Root dove i Certificati X.509 CA NON DEVONO superare il **periodo di validità di 5 anni**.
- **Certificato X.509 dell'Entità Federata**: Ogni partecipante della federazione riceve un certificato X.509 che opera come sub-CA limitata dove i Certificati X.509 NON DEVONO superare il **periodo di validità di 2 anni**.
- **Certificati X.509 del Protocollo**: Certificati X.509 auto-emessi per servizi interni dove i Certificati X.509 NON DOVREBBERO superare il **periodo di validità di 1 anno**.

Ogni entità federata DEVE esporre il proprio Certificato X.509 dell'Entità Federata su un endpoint pubblicamente accessibile. La chiave privata dell'Entità Federata serve a scopi duali:

1. Auto-emissione di Certificati X.509 del Protocollo per operazioni crittografiche interne (capacità sub-CA limitata).
2. Agire come Chiave dell'Entità Federata per la firma delle Dichiarazioni dell'Entità.

Nota

Le Foglie della Federazione possono SOLO emettere Certificati X.509 su se stesse e per scopi specifici dell'applicazione. Le Foglie della Federazione NON DEVONO emettere Certificati X.509 per altre entità federate, come limitato dalle loro Autorità Federali utilizzando l'estensione X.509 Name Constraints.

Per i Certificati X.509 specifici del protocollo, con periodi di validità superiori a 24 ore, l'entità emittente DEVE pubblicare e aggiornare regolarmente una Lista di Revoca dei Certificati X.509 (CRL) su un endpoint pubblicamente accessibile.

9.2 Struttura e Analisi della Catena di Certificati X.509

Le entità federate ricevono catene di Certificati X.509 durante il processo di onboarding. Le entità federate DEVONO validare queste catene su se stesse.

9.2.1 Visualizzazione della Catena di Certificati X.509

Il seguente script consente alle entità federate di:

- Estrarre i dettagli dei certificati X.509 per la verifica.
- Analizzare le estensioni e i vincoli dei certificati X.509.
- Verificare la gerarchia e le relazioni dei certificati X.509

```
#!/bin/bash
# Certificate chain analysis for federation entities
# Array containing certificates in DER format encoded in Base64
certificate_chain=(
    "MIIDyzCCA3GgAwIBAgI..." # Federation Entity Certificate
    "MIIDQzCCAuigAwIBAgI..." # Trust Anchor Certificate
)

# Display first certificate (Federation Entity)
echo "==================================="
echo " Federation Entity Certificate Analysis"
echo "==================================="
echo
echo "${certificate_chain[0]}" | base64 -d > federation_entity.der
```

```
openssl x509 -in federation_entity.der -inform DER -text -noout

# Display second certificate (Trust Anchor)
echo "===================================="
echo " Trust Anchor Certificate Analysis"
echo "===================================="
echo
echo "${certificate_chain[1]}" | base64 -d > trust_anchor.der
openssl x509 -in trust_anchor.der -inform DER -text -noout

# Cleanup temporary files
rm federation_entity.der trust_anchor.der
```

9.2.2 Validazione della Catena di Certificati X.509

Le entità federate DEVONO validare le catene di certificati X.509 per garantire l'instaurazione corretta della fiducia e verificare la conformità con i requisiti PKI della federazione.

Un esempio non normativo della procedura di validazione della catena di Certificati X.509 è fornito di seguito:

```
#!/bin/bash
# Certificate chain validation for federation entities

# Convert DER certificates to PEM format for validation
openssl x509 -inform der -in federation_entity.der -out federation_entity.pem
openssl x509 -inform der -in trust_anchor.der -out trust_anchor.pem

# Validate Trust Anchor certificate (self-signed)
echo "Validating Trust Anchor certificate..."
openssl verify -CAfile trust_anchor.pem trust_anchor.pem

# Validate Federation Entity certificate against Trust Anchor
echo "Validating Federation Entity certificate..."
openssl verify -CAfile trust_anchor.pem federation_entity.pem

# Cleanup
rm federation_entity.pem trust_anchor.pem
```

Le entità federate DOVREBBERO verificare:

1. **Firme dei Certificati X.509**: Ogni Certificato X.509 DEVE essere correttamente firmato dal suo emittente.
2. **Integrità della Catena di Certificati X.509**: Le relazioni Emittente-Soggetto DEVONO essere valide in tutta la catena di Certificati X.509.

3. **Periodi di Validità dei Certificati X.509**: Tutti i Certificati X.509 DEVONO essere entro i loro periodi di validità e DEVONO conformarsi ai limiti della federazione.
4. **Estensioni dei Certificati X.509**: I Vincoli di Base e l'Uso della Chiave DEVONO conformarsi ai requisiti della federazione:
   - Certificati X.509 dell'Entità Federata: `CA:TRUE, pathlen:0` (può solo emettere certificati su se stessa).
   - Certificati X.509 del Protocollo: `CA:FALSE` (non può emettere certificati).

9.3 Gestione della Revoca dei Certificati X.509

Le entità federate DEVONO implementare la verifica di revoca dei Certificati X.509 per garantire la validazione continua della fiducia e la conformità con i requisiti di sicurezza della federazione.

9.3.1 Distribuzione e Accesso CRL

Le autorità federali pubblicano Liste di Revoca dei Certificati X.509 (CRL) su endpoint pubblicamente accessibili. Le entità federate DEVONO essere in grado di accedere e processare queste distribuzioni CRL per la verifica di revoca.

La seguente procedura consente alle entità federate di:

- Localizzare gli endpoint di distribuzione CRL dai Certificati X.509.
- Scaricare le liste di revoca correnti.
- Analizzare il contenuto CRL e i periodi di validità.

```
#!/bin/bash
# CRL extraction and analysis for federation entities

# Extract CRL URL from certificate CRL Distribution Points extension
crl_url=$(openssl x509 -in certificate.der -inform DER -text -noout | \
          grep "URI:" | sed 's/.*URI://')

echo "CRL Distribution Point: $crl_url"

# Download CRL from distribution point
curl -s -O "$crl_url"
crl_file=$(basename "$crl_url")

# Display CRL information
echo "CRL Content Analysis:"
openssl crl -in "$crl_file" -inform DER -text -noout
```

9.3.2 Verifica della Revoca dei Certificati X.509

Le entità federate DEVONO verificare lo stato di revoca dei certificati X.509 controllando i numeri seriali dei certificati X.509 contro le Liste di Revoca dei Certificati X.509 correnti.

Le entità federate DOVREBBERO implementare controlli di revoca automatizzati per:

- **Certificati X.509 dell'Entità Federata**: Verificare periodicamente lo stato del proprio certificato X.509.
- **Certificati X.509 delle Entità Peer**: Validare i Certificati X.509 di altri partecipanti della federazione.
- **Validazione della Catena di Fiducia**: Garantire che intere catene di certificati X.509 rimangano valide.

Di seguito è fornito un esempio non normativo di script bash per la verifica dello stato di revoca dei certificati X.509:

```
#!/bin/bash
# Certificate revocation verification for federation entities

# Extract certificate serial number
certificate_serial=$(openssl x509 -in certificate.der -inform DER -noout -serial | \
                  cut -d= -f2)

# Normalize serial number (remove leading zeros, convert to lowercase)
normalized_serial=$(echo "$certificate_serial" | sed 's/^0*//' | tr '[:upper:]' '[:lower:]')

echo "Certificate Serial Number: $normalized_serial"

# Extract CRL URL and download
crl_url=$(openssl x509 -in certificate.der -inform DER -text -noout | \
          grep "URI:" | sed 's/.*URI://')
curl -s -O "$crl_url"
crl_file=$(basename "$crl_url")

# Validate CRL signature against Trust Anchor
echo "Validating CRL signature..."
openssl crl -in "$crl_file" -inform DER -noout -text -CAfile trust_anchor.pem

# Extract revoked serial numbers from CRL
revoked_serials=$(openssl crl -in "$crl_file" -inform DER -text -noout | \
                 grep 'Serial Number' | \
                 sed 's/.*Serial Number: //' | \
                 sed 's/^0*//' | \
                 tr '[:upper:]' '[:lower:]')
```

```
# Check if certificate is revoked
if echo "$revoked_serials" | grep -q "$normalized_serial"; then
    echo "Certificate Status: REVOKED"
    exit 1
else
    echo "Certificate Status: VALID"
    exit 0
fi
```

9.4 Migliori Pratiche per la Gestione dei Certificati X.509

9.4.1 Integrazione della Validazione dei Certificati X.509

Le entità federate DOVREBBERO integrare le procedure di validazione dei Certificati X.509 nelle loro operazioni standard della federazione:

1. **Aggiornamenti della Configurazione dell'Entità**: Verificare le catene di Certificati X.509 quando si processano gli authority hints e gli aggiornamenti dei Certificati X.509.
2. **Costruzione della Catena di Fiducia**: Validare tutti i Certificati X.509 durante le procedure di costruzione della catena di fiducia.
3. **Operazioni PKI X.509**: Eseguire controlli di revoca dei Certificati X.509 utilizzando gli endpoint CRL.
4. **Gestione dei Certificati del Protocollo**: Validare il Certificato X.509 specifico del protocollo auto-emesso per i servizi interni.
5. **Validazione Periodica**: Implementare programmi regolari di validazione dei Certificati X.509 e CRL.

9.4.2 Diagnostica e Risoluzione dei Problemi

Le entità federate DEVONO implementare procedure diagnostiche per identificare e risolvere problemi relativi ai Certificati X.509:

- **Validazione dei Certificati X.509**, inclusi:
  - **Discrepanze dell'Identificatore della Chiave dell'Autorità**: L'Identificatore della Chiave dell'Autorità CRL non corrisponde all'Identificatore della Chiave del Soggetto del Trust Anchor.
  - **Rotazione del Certificato X.509 del Trust Anchor**: Certificato X.509 del Trust Anchor obsoleto che causa fallimenti di validazione.

  - **Problemi del Formato del Numero Seriale**: Problemi di normalizzazione del numero seriale nel controllo di revoca.

- **Fallimento della Validazione CRL**: Quando la validazione CRL fallisce, le entità federate DOVREBBERO:

  1. **Verificare il Certificato X.509 del Trust Anchor**: Garantire che venga utilizzato il certificato X.509 del Trust Anchor corrente.
  2. **Controllare l'Identificatore della Chiave dell'Autorità**: Confrontare l'Identificatore della Chiave dell'Autorità CRL con l'Identificatore della Chiave del Soggetto del Trust Anchor.
  3. **Validare la Firma CRL**: Verificare che la CRL sia correttamente firmata dall'autorità emittente prevista.
  4. **Aggiornare il Certificato X.509 del Trust Anchor**: Scaricare il certificato X.509 del Trust Anchor aggiornato se è avvenuta una rotazione.

- **Verifica dell'Accessibilità degli Endpoint**: Le entità federate DOVREBBERO implementare test di connettività per gli endpoint dell'infrastruttura dei certificati.

Il seguente esempio non normativo fornisce uno script per il test di connettività dell'infrastruttura dei certificati X.509 della Federazione:

```
#!/bin/bash
# Federation certificate infrastructure connectivity test

# Test Trust Anchor certificate endpoint
ta_cert_url="https://trust-anchor.eid-wallet.example.it/pki/ta.cer"
if curl -f -s "$ta_cert_url" > /dev/null; then
    echo "Trust Anchor certificate endpoint: ACCESSIBLE"
else
    echo "Trust Anchor certificate endpoint: FAILED"
fi

# Test CRL distribution endpoints
ta_crl_url="https://trust-anchor.eid-wallet.example.it/pki/ta.crl"
if curl -f -s "$ta_crl_url" > /dev/null; then
    echo "Trust Anchor CRL endpoint: ACCESSIBLE"
else
    echo "Trust Anchor CRL endpoint: FAILED"
fi
```

9.4.3 Coordinamento del Ciclo di Vita dei Certificati X.509

Le entità federate DEVONO coordinare la gestione dei Certificati X.509 con le procedure del ciclo di vita della federazione seguendo i periodi di validità stabiliti:

- **Rinnovo dei Certificati X.509**: Allineare i rinnovi dei certificati X.509 con gli aggiornamenti della Configurazione dell'Entità e i cicli di aggiornamento dei Trust Mark, secondo i limiti della federazione definiti in *Architettura PKI della Federazione* (pagina 271).
- **Rotazione delle Chiavi**: Coordinare la rotazione della Chiave dell'Entità Federata con le procedure di rinnovo dei certificati X.509.
- **Gestione CRL**: Per i Certificati X.509 del Protocollo con validità > 24 ore, mantenere la pubblicazione CRL corrente.
- **Uscita dalla Federazione**: Garantire la revoca corretta dei Certificati X.509 durante l'uscita volontaria o avviata dall'organo di supervisione dalla federazione.

9.5 Procedure di Rotazione delle Chiavi

Le entità federate DEVONO implementare meccanismi di rotazione delle chiavi per mantenere la postura di sicurezza e rispondere agli eventi di sicurezza. Questa sezione definisce le procedure operative per rilevare e rispondere alle rotazioni delle chiavi superiori e per il rinnovo delle chiavi subordinate.

9.5.1 Rotazione delle Chiavi Superiori

Le entità subordinate DEVONO eseguire la verifica periodica della propria Trust Chain per rilevare le rotazioni delle chiavi superiori e garantire la continuità dell'appartenenza alla federazione.

Requisiti di Verifica

I Subordinati della federazione DEVONO:

1. Recuperare il proprio Subordinate Statement dal superiore immediato ogni 24 ore utilizzando il fetch endpoint definito in *Endpoint API di Federazione* (pagina 123). Ciò DOVREBBE essere fatto entro un intervallo di tempo non superiore a 24 ore.
2. Verificare lo stato di revoca o sospensione.
3. Aggiornare il proprio Entity Configuration quando cambiano i certificati superiori.

Meccanismi di Rilevamento

I Subordinati POSSONO utilizzare uno o più dei seguenti meccanismi per rilevare le modifiche:

- **Verifica CRL/OCSP**: Monitorare le Certificate Revocation List o le risposte del protocollo Online Certificate Status Protocol per rilevare la revoca dei certificati superiori come definito in *Revoca di Certificati X.509* (pagina 155). La revoca del certificato indica che è in corso la rotazione della chiave superiore.
- **Fetch Endpoint**: Recuperare il Subordinate Statement tramite `GET {superiore}/fetch?sub={entity_id_subordinato}` per ottenere i certificati correnti.
- **Events Endpoint**: Monitorare `GET {superiore}/federation_subordinate_events_endpoint?sub={entity_id_subordinato}` per gli eventi `jwks_update`. Il Federation Subordinate Events endpoint restituisce la traccia storica degli eventi di registrazione come definito in *Endpoint API di Federazione* (pagina 123).

I Subordinati DOVREBBERO implementare la pianificazione automatizzata per:

- Operazioni di fetch giornaliere per recuperare il Subordinate Statement corrente.
- Validazione della catena di certificati X.509 dopo ogni fetch.
- Aggiornamenti automatici dell'Entity Configuration quando vengono rilevate modifiche ai certificati.
- Monitoraggio degli eventi per il rilevamento proattivo delle modifiche.

Quando un'entità superiore (Trust Anchor o Intermediate) ruota la propria Chiave dell'Entità Federata, tutti i certificati X.509 Subordinati DEVONO essere riemessi.

Procedura di Rotazione

1. **Revoca della Chiave**: Il superiore pubblica la Certificate Revocation List (CRL) o la voce Historical Keys (HK) revocando la vecchia chiave come definito in *Revoca di Certificati X.509* (pagina 155).
2. **Aggiornamento Entity Configuration**: Il superiore pubblica la nuova Entity Configuration firmata con la nuova Chiave dell'Entità Federata.
3. **Riemissione Certificati X.509**: Il superiore riemette i certificati X.509 a tutti i Subordinati utilizzando la nuova chiave.
4. **Pubblicazione Certificati**: Il superiore rende disponibili i nuovi certificati sul fetch endpoint.
5. **Rilevamento Subordinato**: I Subordinati rilevano la modifica al successivo fetch programmato (al massimo entro 24 ore dalla pubblicazione della modifica).
6. **Aggiornamento Certificati**: I Subordinati aggiornano il campo `x5c` nella propria Entity Configuration con il nuovo certificato.

Priorità dei Certificati

Quando esistono più certificati X.509 per la stessa chiave pubblica subordinata, il certificato nel Subordinate Statement DEVE avere priorità sul certificato nella Entity Configuration del subordinato.

Nota

Un'entità superiore non può avviare autonomamente un processo di rotazione delle Chiavi Subordinate. Il meccanismo di rotazione delle Chiavi Subordinate è sempre avviato dall'entità Subordinata, vedi Sezione *Rotazione delle Chiavi Subordinate* (pagina 280).

9.5.2 Rotazione delle Chiavi Subordinate

I Subordinati ruotano le proprie Chiavi dell'Entità Federata inviando Certificate Signing Request al proprio superiore immediato seguendo la stessa procedura dell'onboarding iniziale.

Procedura di Rotazione

1. **Generazione Nuova Chiave**: Il subordinato genera una nuova coppia di Chiavi dell'Entità Federata.
2. **Generazione CSR**: Il subordinato genera un Certificate Signing Request PKCS #10 per la nuova chiave come definito in *Onboarding delle Entità* (pagina 239).
3. **Invio CSR**: Il subordinato invia il CSR all'entità superiore utilizzando lo stesso endpoint di onboarding.
4. **Emissione Certificato**: Il superiore valida il CSR ed emette un nuovo certificato X.509.
5. **Pubblicazione Certificato**: Il superiore rende disponibile il nuovo certificato sul fetch endpoint.
6. **Recupero Certificato**: Il subordinato recupera il Subordinate Statement aggiornato contenente il nuovo certificato.
7. **Aggiornamento Entity Configuration**: Il subordinato aggiunge la nuova JWK con il nuovo certificato all'array `jwks` della Entity Configuration.

Nota

Durante la rotazione delle chiavi, i Subordinati DOVREBBERO mantenere sia le chiavi vecchie che quelle nuove per garantire la continuità del servizio. La vecchia chiave DOVREBBE essere rimossa solo dopo aver confermato la corretta propagazione della nuova chiave in tutta la federazione.

Gestione del Ciclo di Vita dell'Entità

Questa sezione fornisce le procedure di implementazione tecnica per la gestione del ciclo di vita dell'entità. Per i concetti di alto livello del ciclo di vita e i processi aziendali, vedere *Gestione del Ciclo di Vita delle Entità* (pagina 30).

Mentre l'aggiornamento dei dati amministrativi DEVE seguire processi di governance che sono fuori dall'ambito di questa specifica tecnica, gli aggiornamenti della configurazione tecnica sono descritti nelle seguenti sezioni.

9.6 Aggiornamenti della Configurazione Tecnica

Gli aggiornamenti tecnici che influenzano le operazioni del protocollo della federazione DEVONO seguire procedure specifiche per:

- **Rinnovo dei Certificati X.509**
  1. **Preparazione Pre-rinnovo**: L'Entità DEVE generare un nuovo CSR con informazioni aggiornate del certificato X.509.
  2. **Richiesta di Rinnovo**: L'Entità DEVE inviare la richiesta di rinnovo con il nuovo CSR seguendo la stessa procedura tecnica dell'onboarding iniziale.
  3. **Integrazione del Certificato X.509**: L'Entità DEVE aggiornare la sua Configurazione dell'Entità con la nuova catena di certificati X.509 nel parametro `x5c`.
  4. **Validazione della Catena di Fiducia**: L'Entità DEVE verificare la Catena di Fiducia aggiornata attraverso l'endpoint `/resolve`.
  5. **Aggiornamento del Registro**: L'Entità DOVREBBE confermare le informazioni dell'entità aggiornate nell'endpoint `/list` del Trust Anchor.
- **Rotazione delle Chiavi**
  1. **Attivazione della Chiave**: In caso di incidente o rotazione pianificata, l'Entità DEVE attivare una Chiave dell'Entità di Federazione alternativa per le operazioni di firma.
  2. **Generazione di Nuova Chiave**: L'Entità DEVE generare una nuova coppia di Chiave Pubblica dell'Entità di Federazione per servire come chiave aggiuntiva.
  3. **Pubblicazione Parallela delle Chiavi**: L'Entità DEVE pubblicare tutte le chiavi disponibili nella claim `jwks` della Configurazione dell'Entità durante il periodo di transizione.

4. **Richiesta di Certificato X.509**: L'Entità DEVE richiedere un nuovo certificato X.509 per la nuova chiave seguendo la procedura standard.
5. **Migrazione Graduale**: L'Entità DEVE aggiornare la Configurazione dell'Entità per utilizzare la chiave attivata per la firma mantenendo tutte le chiavi per la verifica.
6. **Deprecazione della Chiave Vecchia**: L'Entità DEVE rimuovere la vecchia chiave dalla Configurazione dell'Entità dopo il periodo di validazione, mantenendo la chiave alternativa e la nuova chiave.

- **Aggiornamenti dei Metadati**
  - **Modifiche degli Endpoint**: L'Entità PUÒ aggiornare gli endpoint di servizio nei metadati specifici dell'entità.
  - **Aggiornamenti delle Capacità**: L'Entità PUÒ modificare i protocolli supportati, gli algoritmi o le capacità di servizio entro i vincoli definiti da questo profilo di implementazione IT-Wallet.

Tutti gli aggiornamenti tecnici DEVONO essere validati attraverso:

1. **Validazione della Configurazione dell'Entità**: L'Entità DEVE verificare la struttura e il contenuto della EC aggiornata.
2. **Risoluzione della Catena di Fiducia**: L'Entità DEVE confermare che l'endpoint `/resolve` restituisca una Catena di Fiducia valida.
3. **Stato della Federazione**: L'Entità DEVE verificare lo stato operativo dell'entità nel registro della federazione.
4. **Test di Integrazione**: L'Entità DOVREBBE testare le operazioni del protocollo della federazione con la configurazione aggiornata.

9.7 Procedure Tecniche di Uscita dalla Federazione

Per il contesto aziendale sui processi di uscita dalla federazione, vedere *Processi di Uscita e Rimozione dalla Federazione* (pagina 31). Questa sezione copre le procedure di implementazione tecnica.

9.7.1 Uscita Volontaria - Disattivazione Tecnica

1. **Richiesta di Revoca del Certificato X.509**: L'Entità DEVE inviare una richiesta di revoca del Certificato X.509 all'Autorità Federata con il motivo della revoca. La richiesta DEVE essere firmata con la Chiave Privata dell'Entità Federata corrispondente al Certificato X.509 che viene revocato per dimostrare la legittimità della richiesta di revoca.
2. **Verifica dell'Aggiornamento CRL**: L'Autorità Federata DEVE revocare il Certificato X.509 dell'Entità e l'Entità DEVE verificare che appaiano nella Lista di Revoca dei Certificati X.509 aggiornata.
3. **Rimozione della Dichiarazione Subordinata**: L'Autorità Federata DEVE rimuovere completamente la Dichiarazione Subordinata dell'Entità dal Registro della Federazione per prevenire qualsiasi validazione del rapporto di fiducia, e quindi rimuovere qualsiasi riferimento a quella Entità nell'endpoint di elenco, endpoint di risoluzione o endpoint di elenco con trust mark.
4. **Disattivazione della Configurazione dell'Entità**: L'Entità DEVE disattivare la sua Configurazione dell'Entità. L'Entità PUÒ:
    a. Rimuovere completamente la Configurazione dell'Entità dall'endpoint `/.well-known/openid-federation` (restituendo HTTP 404), OPPURE
    b. Mantenere la Configurazione dell'Entità come scaduta (con la claim `exp` nel passato). Pertanto NON DEVE aggiornarla con timestamp freschi.
5. **Aggiornamento dello Stato del Registro**: L'Entità DOVREBBE verificare la rimozione dal Registro della Federazione, verificando anche lo stato del Trust Mark utilizzando l'endpoint dello Stato del Trust Mark.

Esempio non normativo di richiesta di revoca del Certificato X.509 seguendo il formato **RFC 3280**:

```
Richiesta di Revoca del Certificato X.509:
Soggetto: CN=credentials.example.gov, OU=Digital Credentials, O=Example Organization, L=Roma,
ST=Lazio, C=IT, emailAddress=technical@credentials.example.gov
Numero Seriale del Certificato X.509: 987654321
Motivo della Revoca: cessation_of_operation (5)
Data di Revoca: 2025-12-31T23:59:59Z
```

```
Richiesta firmata con la Chiave Privata dell'Entità Federata corrispondente a:
Algoritmo della Chiave Pubblica: id-ecPublicKey
OID ASN1: prime256v1
CURVA NIST: P-256
ID Chiave: NsXymfIILEPR5Y0t

Nota: La CRR DEVE essere firmata con la stessa chiave privata che corrisponde al
certificato X.509 che viene revocato per autenticare la richiesta di revoca.
```

Esempio CRR in formato DER (codificato Base64):

```
-----BEGIN CERTIFICATE REVOCATION REQUEST-----
MIIBVjCB/wIBADBpMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNV
BAcMBFJvbWExGjAYBgNVBAoMEUV4YW1wbGUgT3JnYW5pemF0aW9uMR8wHQYDVQQD
DBZjcmVkZW50aWFscy5leGFtcGxlLmdvdjBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABIBgZ4HBgUCNXwY5LJSlKzm7gXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq
0ob4l_gslT14644zbYXYF-xmw7aPdlbMuw3T1URwI4nafMtKrYwDQYJKoZIhvcNAQ
kEAwIJrRLl1VR987654321gBgJKwYBBQUHAgEWHGh0dHBzOi8vZXhhbXBsZS5vcmcv
cG9saWN5MAoGCCqGSM49BAMCA0gAMEUCIQC9h3Y6hFgd7zUzZyBrQ3jJ8HmVF2Qa
-----END CERTIFICATE REVOCATION REQUEST-----
```

9.7.2 Rimozione da Parte dell'Organo di Supervisione - Implementazione Tecnica

1. **Revoca di Emergenza del Certificato X.509**: L'Autorità Federata DEVE revocare immediatamente il Certificato X.509 con il codice di motivo appropriato (ad es., "Key Compromise", "Cessation of Operation").
2. **Aggiornamento di Emergenza CRL**: Il Trust Anchor DEVE pubblicare la CRL aggiornata entro il tempo di emergenza.
3. **Rimozione della Dichiarazione Subordinata**: L'Autorità Federata DEVE interrompere immediatamente l'emissione delle Dichiarazioni Subordinate dell'Entità utilizzando l'endpoint fetch, e/o qualsiasi riferimento a quella Entità utilizzando l'endpoint di elenco, o l'endpoint di elenco con trust mark, o l'endpoint di risoluzione.
4. **Invalidazione della Configurazione dell'Entità**: La Configurazione dell'Entità in `/.well-known/openid-federation` diventa invalida a causa della revoca del Certificato X.509 (la verifica della firma fallisce).
5. **Invalidazione della Catena di Fiducia**: La risoluzione della Catena di Fiducia DEVE restituire uno stato di errore per l'entità interessata.
6. **Isolamento degli Endpoint di Servizio**: L'infrastruttura della federazione DEVE bloccare l'accesso agli endpoint di servizio della federazione.

Esempio di verifica di revoca di emergenza:

```
# Controlla l'aggiornamento di emergenza CRL
curl -o emergency.crl https://trust-anchor.eid-wallet.example.it/pki/ta-sub.crl
openssl crl -in emergency.crl -text -noout | grep "Last Update"

# Verifica che la risoluzione della Catena di Fiducia fallisca
curl "https://trust-anchor.eid-wallet.example.it/resolve?sub=https%3A//suspended-entity.example
.it&trust_anchor=https%3A//trust-registry.eid-wallet.example.it"
# Previsto: HTTP 404 o risposta di errore specifica
```

Modifiche Tecniche a Livello di Componente

Componenti tecnici specifici POSSONO essere modificati mantenendo l'appartenenza alla federazione:

```
{
  "iss": "https://ci.example.it",
  "sub": "https://ci.example.it",
  "jwks": {
    // contenuto jwks
  },
  "metadata": {
    "openid_credential_issuer": {
      "credential_endpoint": "https://ci.example.it/credential",
      "credentials_supported": [ ]
      // rimosso: "batch_credential_endpoint" per manutenzione
    }
  }
}
```

L'Entità DEVE seguire questi passaggi per le modifiche dei componenti:

1. **Aggiornamento della Configurazione dell'Entità**: L'Entità DEVE modificare i metadati per riflettere le modifiche dei componenti.
2. **Rivalidazione della Catena di Fiducia**: L'Entità DEVE verificare la configurazione aggiornata attraverso l'endpoint `/resolve`.
3. **Test del Servizio**: L'Entità DOVREBBE testare le operazioni del protocollo della federazione rimanenti.
4. **Verifica del Registro**: L'Entità DOVREBBE confermare le capacità aggiornate nel registro della federazione.

Obblighi Tecnici Post-Uscita

Le entità che escono dalla federazione DEVONO mantenere quanto segue per la conformità normativa:

1. **Configurazione Storica dell'Entità**: L'Entità DEVE mantenere l'accessibilità dell'endpoint `/.well-known/openid-federation` per scopi di audit.

2. **Archivio della Catena di Certificati X.509**: L'Entità DEVE mantenere le catene di Certificati X.509 accessibili per la verifica delle Credenziali esistenti (minimo 7 anni).
3. **Conservazione dei Log di Audit**: L'Entità DEVE archiviare i log del protocollo della federazione per i requisiti normativi.

CAPITOLO 10

Entità

10.1 Soluzione Wallet

La Wallet Solution è emessa dal Fornitore di Wallet sotto forma di applicazione mobile e servizi, come interfacce web. L'applicazione mobile funge da interfaccia principale per gli Utenti, consentendo loro di conservare i propri Attestati Elettronici e di interagire con altri partecipanti dell'ecosistema, come i Fornitori di Attributi Elettronici e le Relying Party. Questi Attestati Elettronici costituiscono un insieme di dati che possono identificare in modo univoco una persona fisica o giuridica, insieme ad altre Attestazioni Elettroniche Qualificate e non Qualificate di Attributi, note rispettivamente come QEAAs ed EAAs, o più brevemente (Q)EAAs. Una volta che un utente installa l'applicazione mobile sul proprio dispositivo, tale installazione viene definita come istanza del Wallet per quell'utente. Supportando l'applicazione mobile, il Fornitore di Wallet garantisce la sicurezza e l'affidabilità dell'intera Wallet Solution, poiché è responsabile dell'emissione della Wallet Instance Attestation (WIA) e della Key Attestation (KA). La WIA dimostra l'autenticità e l'integrità dell'istanza del Wallet, mentre la KA fornisce la prova che le chiavi utilizzate per il collegamento crittografico delle credenziali sono archiviate in modo sicuro all'interno di un WSCD affidabile. Inoltre, la KA conferma che la Wallet Unit non è stata revocata.

Il seguente diagramma illustra l'Architettura di Alto Livello della Soluzione Wallet.

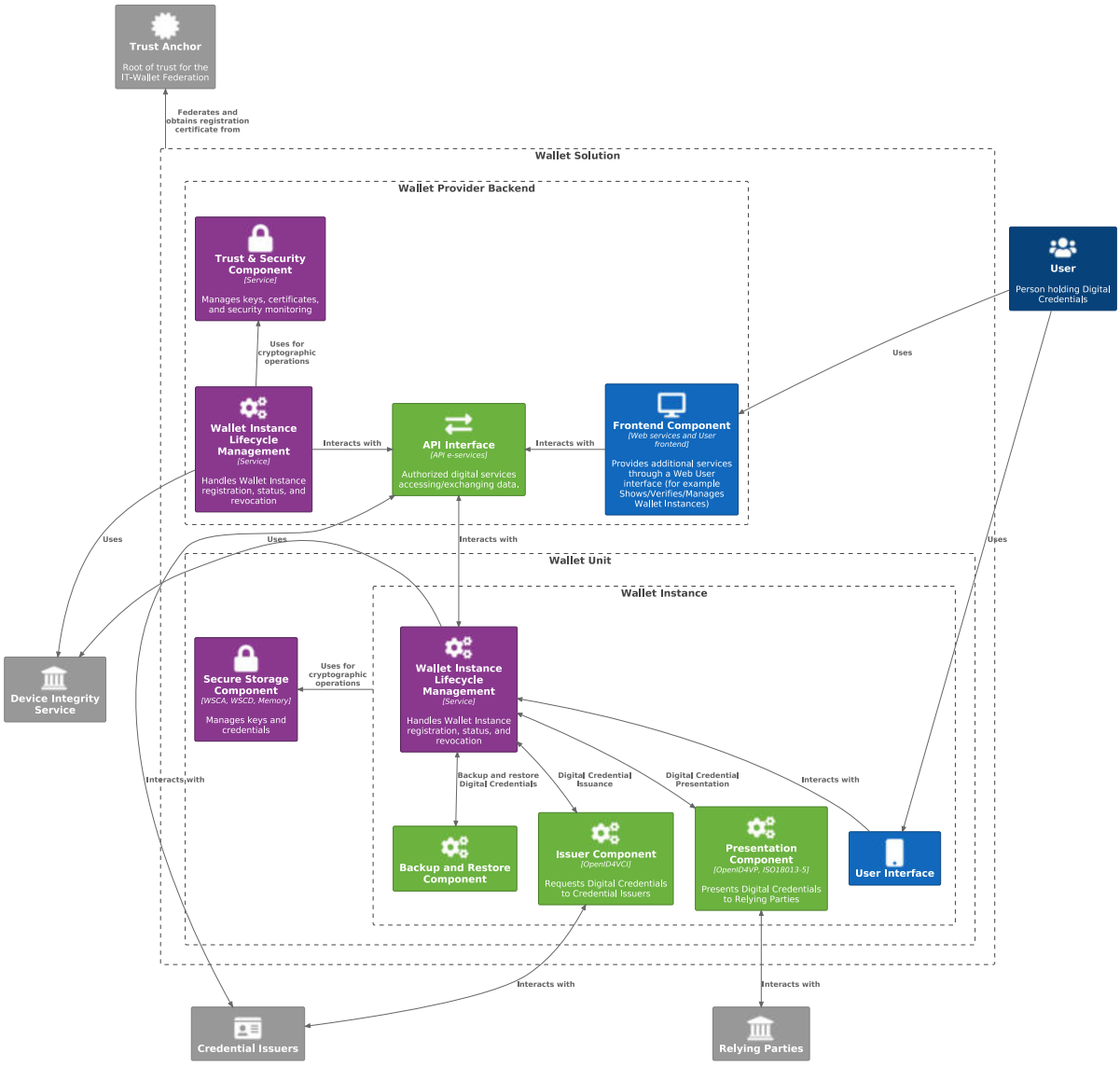


Fig. 10.1: Architettura di Alto Livello della Soluzione Wallet.

10.1.1 Requisiti della Soluzione Wallet

Questa sezione elenca i requisiti relativi ai Fornitori di Wallet e alle Soluzioni Wallet con le loro Istanze del Wallet, nonché i corrispondenti Wallet Instance Attestation, Key Attestation e il componente di secure storage (WSCD o Keystore).

- La Soluzione Wallet DEVE aderire alle specifiche stabilite da questo documento per ottenere Attestati Elettronici di Dati di Identificazione Personale (PID) e Attestati Elettronici di Attributi (Q)EAA.
- Il Fornitore di Wallet DEVE esporre un insieme di endpoint, disponibili esclusivamente per le istanze della sua Soluzione Wallet, che supportano le funzionalità principali delle Istanze del Wallet.
- L'Istanza del Wallet DEVE periodicamente ristabilire la trust con il suo Fornitore di Wallet, ottenendo una nuova Wallet Instance Attestation (*WP_018* (pagina 781)).
- L'istanza del Wallet DEVE stabilire un rapporto di fiducia con gli altri partecipanti dell'ecosistema del Wallet, come i Fornitori di Attributi Elettronici. Nel caso dei Fornitori di Attributi Elettronici, l'istanza del Wallet presenta sia la Wallet Instance Attestation che la Key Attestation.
- L'Istanza del Wallet DEVE essere compatibile e funzionale sia sui sistemi operativi Android che iOS e disponibile rispettivamente sul Play Store e sull'App Store (*WP_015* (pagina 781)).
- L'Istanza del Wallet DEVE fornire un meccanismo per verificare l'effettivo possesso e il pieno controllo da parte dell'Utente del proprio dispositivo personale.
- L'Istanza del Wallet DEVE fornire agli Utenti un elenco aggiornato delle Relying Party con cui l'Utente ha stabilito una connessione e, ove applicabile, tutti i dati scambiati;
- L'Istanza del Wallet DEVE fornire agli Utenti un meccanismo per richiedere la cancellazione degli attributi personali da parte di una Relying Party ai sensi dell'articolo 17 del Regolamento (UE) 2016/679, e per registrare ogni Richiesta di Cancellazione effettuata.

Nota

Non esiste una corrispondenza stretta uno-a-uno tra i requisiti in questa sezione e i casi di test in *Matrice di Test per Wallet Provider* (pagina 778). Alcuni requisiti sono espressi a un livello troppo alto per poter essere rappresentati come casi di test atomici, mentre altri sono già affrontati in modo più dettagliato all'interno dei flussi correlati (ad es. *Emissione della Wallet Instance Attestation* (pagina 305)).

Requisiti della Wallet Instance Attestation

la Wallet Instance Attestation contiene informazioni riguardanti il livello di sicurezza del dispositivo che ospita l'Istanza del Wallet. Esso dimostra principalmente l'**autenticità**, l'**integrità**, la **sicurezza** e in generale l'**affidabilità** di una particolare Istanza del Wallet.

I requisiti per la Wallet Instance Attestation sono definiti di seguito:

- la Wallet Instance Attestation DEVE fornire tutte le informazioni rilevanti per attestare l'**integrità** e la **sicurezza** del dispositivo in cui è installata l'Istanza del Wallet (*WP_019* (pagina 781)).
- la Wallet Instance Attestation DEVE essere firmato dal Fornitore di Wallet che ha autorità e proprietà sulla Soluzione Wallet, come specificato dalla Registration Authority di supervisione. Questo garantisce che la Wallet Instance Attestation colleghi in modo univoco il Fornitore di Wallet a questa particolare Istanza del Wallet (*WP_020* (pagina 781)).
- Il Fornitore di Wallet DEVE periodicamente valutare e garantire l'integrità, l'autenticità e la genuinità dell'Istanza del Wallet. Il Fornitore di Wallet verifica l'Istanza del Wallet utilizzando il flusso più sicuro reso disponibile dalle API del Fornitore del Sistema Operativo, come la *Play Integrity API* per Android e *App Attest* per iOS (*WP_011* (pagina 778)).
- la Wallet Instance Attestation DEVE essere vincolato in modo sicuro alla chiave pubblica effimera dell'Istanza del Wallet (*WP_019b* (pagina 781)).
- la Wallet Instance Attestation PUÒ essere utilizzato più volte durante il suo periodo di validità, consentendo autenticazioni e autorizzazioni ripetute senza la necessità di richiedere nuovi attestati ad ogni interazione. Tuttavia, è RACCOMANDATO che le Istanze del Wallet evitino di utilizzare ripetutamente lo stesso attestato, a causa di preoccupazioni sulla privacy come la possibilità di collegamento tra diverse interazioni.
- La Wallet Instance Attestation DEVE avere un time-to-live inferiore a 24 ore: la differenza tra il tempo di scadenza (`exp`) e il tempo di emissione (`iat`) DEVE essere inferiore a 24 ore, oltre il quale NON DEVE più essere considerata valida (*WP_028* (pagina 781), *WP_144* (pagina 814)).
- la Wallet Instance Attestation NON DEVE essere rilasciato dal Fornitore di Wallet se l'autenticità, l'integrità e la genuinità dell'Istanza del Wallet che lo richiede non possono essere garantite (*WP_019a* (pagina 781)).
- Ogni Istanza del Wallet DOVREBBE essere in grado di richiedere più Wallet Instance Attestation utilizzando diverse chiavi pubbliche crittografiche associate ad essi.
- la Wallet Instance Attestation NON DEVE contenere informazioni sull'Utente che controlla l'Istanza del Wallet (*WP_029b* (pagina 781)).

- L'Istanza del Wallet DEVE ottenere una Wallet Instance Attestation come prerequisito per passare allo stato Operativo, come definito da EIDAS-ARF.
- Un Wallet Provider DEVE garantire che una Wallet Unit non revocata presenti in ogni momento una Wallet Instance Attestation temporalmente valida e non revocata a un PID Provider o a un Attestation Provider durante il processo di emissione di un PID o di un'attestazione. Nota: questo requisito si applica sia alle attestazioni associate a un dispositivo che a quelle non associate a un dispositivo, come definito da EIDAS-ARF.
- Una Wallet Unit DEVE presentare una Wallet Instance Attestation esclusivamente a un PID Provider o a un Attestation Provider, nell'ambito del processo di emissione di un PID o di un'attestazione, e non a una Relying Party o a qualsiasi altra entità.

Nota

In questa sezione, i servizi utilizzati per attestare la genuinità dell'Istanza del Wallet e del dispositivo in cui è installata sono indicati come **API del Servizio di Integrità del Dispositivo**. L'API del Servizio di Integrità del Dispositivo è considerata in modo astratto e si presume sia un servizio fornito da una terza parte affidabile (cioè, l'API del Fornitore del Sistema Operativo) in grado di eseguire controlli di integrità sull'Istanza del Wallet e sul dispositivo in cui è installata.

Requisiti della Key Atestation

La Key Attestation contiene informazioni che garantiscono che le chiavi utilizzate per il collegamento crittografico degli Attestati Elettronici siano archiviate in un WSCD o in un Keystore **affidabile**. Inoltre, fornisce un metodo per autenticare quel componente di memorizzazione presso il Credential Issuer e verifica che la Wallet Unit non sia stata revocata.

I requisiti per la Key Attestation sono definiti di seguito:

- La Key Attestation DEVE fornire al PID Provider o all'Attestation Provider informazioni sulle capacità del WSCA e del WSCD della Wallet Unit, in modo che possano prendere una decisione ben fondata sull'opportunità di emettere un PID o un'attestazione per tale Wallet Unit.
- La Key Attestation DEVE consentire ai PID Provider e agli Attestation Provider di verificare l'autenticità e lo stato di revoca della Wallet Unit.
- Un Wallet Provider DEVE garantire che una Wallet Unit non revocata possa in ogni momento presentare una Key Attestation, quando richiesto da un PID Provider o da un Attestation Provider.
- Durante l'emissione di un PID a Livello di Garanzia Alto, la Wallet Unit DEVE fornire al PID Provider una Key Attestation (KA) valida che descriva il WSCD che ha generato la

nuova chiave privata del PID. Le chiavi del PID a LoA High DEVONO essere generate e memorizzate solo in un WSCD. Il profilo implementativo attuale non include l'emissione di PID High; le chiavi dell'IT-Wallet ID e delle (Q)EAA POSSONO essere generate e memorizzate in un Keystore.

- Durante l'emissione di un'attestazione vincolata al dispositivo, una Wallet Unit DEVE recuperare dai metadati dell'Emittente (come specificato in OpenID4VCI) i requisiti dell'Attestation Provider riguardanti l'archiviazione delle chiavi da parte del WSCA/WSCD o del Keystore. La Wallet Unit DEVE determinare quale dei propri WSCA/WSCD o Keystore, se presente, soddisfi tali requisiti. Se un WSCA/WSCD o un Keystore conforme è disponibile per la Wallet Unit, quest'ultima DEVE fornire all'Attestation Provider una KA valida che descriva il WSCA/WSCD o il Keystore selezionato. Nota: una KA descrive le proprietà del WSCA/WSCD o di un Keystore e contiene una o più chiavi pubbliche corrispondenti a chiavi private generate e archiviate in tale WSCA/WSCD o Keystore.
- Se una Wallet Unit contiene più WSCA, essa DEVE, in modo interno e sicuro, tenere traccia di quali PID e attestazioni sono associati a ciascun WSCA.
- Una Wallet Unit DEVE presentare una Key Attestation solo come parte del processo di emissione di un PID o di un'attestazione.
- La Key Attestation DEVE consentire ai PID Provider di richiedere a un Wallet Provider la revoca di una Wallet Unit, includendo un identificatore per la Wallet Unit all'interno della KA (ad esempio, un URI e un indice a una Attestation Status List). Il Wallet Provider DEVE garantire che tale identificatore della Wallet Unit non consenta il tracciamento dell'utente.
- La Key Attestation DEVE contenere una o più chiavi pubbliche di credenziali attestate provenienti dallo stesso WSCD o Keystore.
- La Key Attestation DEVE essere firmata dal Wallet Provider che ha autorità e proprietà sulla Wallet Solution, come specificato dall'Autorità di Registrazione di riferimento. I Wallet Provider DEVONO garantire che i certificati utilizzati per firmare le KA e le WIA siano conformi a tutti i requisiti applicabili della ETSI TS 119 412-6, in particolare alla Clausola 5.
- Un Attestation Provider che emette attestazioni non vincolate al dispositivo DEVE indicare nei propri metadati di Credential Issuer che non necessita di una KA. Una Wallet Unit NON DEVE inviare una KA a un Attestation Provider quando richiede un'attestazione non vincolata al dispositivo. Nota: una Wallet Unit invia una WIA all'Attestation Provider indipendentemente dal fatto che le attestazioni da esso emesse siano vincolate al dispositivo o meno.
- Un Wallet Provider DEVE garantire che la presentazione di una KA sia crittograficamente vincolata allo specifico contesto in cui è destinata a essere utilizzata. Nota: come specificato in OpenID4VCI, ciò si ottiene facendo sì che la KA firmata contenga essa stessa un nonce

fornito dal PID Provider o dall'Attestation Provider durante il processo di emissione. In alternativa, la Wallet Unit presenta la KA insieme a una Proof-of-Possession costituita da una firma su tale nonce, creata dalla chiave privata corrispondente a una delle chiavi pubbliche attestate nella KA.

- Durante l'emissione di un PID o di un'attestazione vincolata al dispositivo, il PID Provider o l'Attestation Provider DEVE verificare la KA in conformità ai requisiti dell'Appendice F.4 di OpenID4VCI.
- Durante l'emissione di un PID o di un'attestazione vincolata al dispositivo, il PID Provider o l'Attestation Provider DEVE ricevere una prova che la Wallet Unit possiede le chiavi private corrispondenti a tutte le chiavi pubbliche presenti nella KA.
- Il WSCA, il WSCD o il Keystore NON DEVONO consentire l'esportazione in chiaro delle chiavi private dell'Utente. Se un dispositivo segnala una chiave privata come esportabile, il Fornitore di Wallet DEVE rifiutare la Key Attestation e NON DEVE attivare l'Istanza del Wallet (*WP_014b* (pagina 781)).
- Un Wallet Provider DEVE considerare tutti i fattori rilevanti, inclusi l'uso offline, l'interoperabilità e il rischio che una KA diventi un vettore per tracciare l'Utente, nel decidere il periodo di validità di una KA.
- La Key Attestation NON DEVE essere emessa dal Wallet Provider se l'affidabilità del WSCD o del Keystore non è garantita. In tal caso, l'Istanza del Wallet DEVE essere revocata.

Requisiti WSCD e Keystore

Per garantire la massima sicurezza, le chiavi crittografiche associate a un'Istanza del Wallet (ad esempio, utilizzate per generare la Wallet Instance Attestation) DEVONO essere generate e memorizzate in modo sicuro in un **Keystore** o in un **WSCD**, come definiti in *Definizioni e Acronimi* (pagina 675). L'Utente DEVE mantenere il controllo esclusivo di tali chiavi private (Sole Control). Il Keystore o il WSCD DEVE richiedere l'autenticazione dell'Utente (sblocco del Wallet: PIN o biometrico) prima di qualsiasi firma o altra operazione con la chiave privata (*WP_014c* (pagina 781)).

Possono essere utilizzati i seguenti approcci:

- **Keystore Interno Locale**: memorizzazione delle chiavi hardware-backed nativa del dispositivo dell'Utente, come il Secure Enclave su iOS, o il Trusted Execution Environment (TEE) e StrongBox su Android. È il componente denominato Local Internal WSCD in alcuni testi ARF. È un Keystore, non un WSCD certificabile High.
- **WSCD Esterno Locale**: hardware esterno al dispositivo dell'Utente, come una smart card conforme a *GlobalPlatform* e che supporta *JavaCard*, certificabile come WSCD.

- **WSCD Remoto**: un Hardware Security Module (HSM) remoto certificabile come WSCD.
- **WSCD Ibrido Locale**: un componente hardware interno collegabile all'interno del dispositivo dell'Utente, come un *eUICC* che aderisce agli standard *GlobalPlatform* e supporta *JavaCard*.
- **WSCD Ibrido Remoto**: un componente locale combinato con un servizio remoto.

Le chiavi private di un IT-Wallet ID e di una (Q)EAA POSSONO essere memorizzate in un Keystore Interno Locale (eIDAS Substantial). Le chiavi private di un PID a LoA High DEVONO essere memorizzate solo in un WSCD. Un Keystore Interno Locale NON DEVE essere trattato come prova di LoA High. Le chiavi private vincolate a una WIA DEVONO essere generate e memorizzate nello stesso Keystore o WSCD che la WIA attesta (*WP_014* (pagina 781)–*WP_014e* (pagina 781)).

Un'Istanza di Relying Party Mobile che non memorizza chiavi di identità dell'Utente PUÒ usare un Keystore Interno Locale e non è tenuta a usare un WSCD certificabile High (*WP_014f* (pagina 781)).

Gli Attestati Elettronici le cui chiavi private vincolate sono memorizzate in un Keystore locale o in un WSCD locale POSSONO essere presentati in prossimità o comunque offline. Gli Attestati Elettronici le cui chiavi private vincolate sono memorizzate in un WSCD remoto NON DEVONO essere presentati offline (*WP_160* (pagina 781)).

Avvertimento

Nella fase attuale, il profilo di implementazione definito in questo documento supporta solo il **Keystore Interno Locale** (*WP_014* (pagina 781)). Le versioni future di questa specifica POTREBBERO includere altri approcci a seconda del Livello di Garanzia dell'Autenticatore richiesto (*AAL*).

Per informazioni più dettagliate, fare riferimento a *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304), *Emissione della Wallet Instance Attestation* (pagina 305) e *Emissione della Key Attestation* (pagina 308) di questo documento.

10.1.2 Componenti della Soluzione Wallet

Nota

I test relativi ai componenti della Wallet Solution (Backend del Wallet e Unità di Wallet) sono riassunti in *WP_012* (pagina 778) e in *WP_013* (pagina 781), rispettivamente.

Backend del Wallet

Componente Frontend

Il Componente Frontend DEVE fornire un'interfaccia Utente basata sul web per la gestione dell'Istanza del Wallet, offrendo funzionalità per:

- Visualizzare e verificare le Istanze del Wallet e il loro stato.
- Gestire il ciclo di vita dell'Istanza del Wallet (ad esempio, revoca).
- Fornire supporto e documentazione all'Utente.

Interfaccia API

Questo componente DEVE:

- inoltrare la richiesta dal Componente Frontend o dall'Istanza del Wallet al componente di Gestione del Ciclo di Vita dell'Istanza del Wallet.
- utilizzare PDND secondo le regole nella Sezione *e-Service PDND* (pagina 714) per essere notificato dal Provider di PID/IT-Wallet ID della necessità di revocare l'Istanza del Wallet e cancellare l'account dell'Utente a causa del decesso dell'Utente.

Gestione del Ciclo di Vita dell'Istanza del Wallet

Questo componente DEVE gestire:

- Registrazione dell'Istanza del Wallet (dettagliata in *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304)).
- Emissione della Wallet Instance Attestation (dettagliata in *Emissione della Wallet Instance Attestation* (pagina 305)).
- Emissione della Key Attestation (dettagliata in *Emissione della Key Attestation* (pagina 308)).
- Gestione dello stato (mantenimento e aggiornamento della validità).
- Processi di revoca (implementazione di meccanismi per revocare le Istanze del Wallet), secondo la Sezione *Revoca dell'Istanza del Wallet* (pagina 312).

Componente Trust & Security

Questo componente DEVE garantire la sicurezza attraverso:

- Gestione di chiavi e certificati.
- Registrazione degli audit.
- Monitoraggio della sicurezza e risposta agli incidenti.
- Conformità ai requisiti di sicurezza della Federazione IT-Wallet.

Unità di Wallet

Interfaccia Utente

L'Interfaccia Utente è il punto di interazione e comunicazione tra l'Utente e l'Istanza del Wallet.

Componente di Gestione del Ciclo di Vita dell'Istanza del Wallet

Interagendo con il Backend del Wallet, questo componente DEVE gestire:

- Registrazione dell'Istanza del Wallet (dettagliata in *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304)).
- Emissione della Wallet Instance Attestation (dettagliata in *Emissione della Wallet Instance Attestation* (pagina 305)).
- Emissione della Key Attestation (dettagliata in *Emissione della Key Attestation* (pagina 308)).
- Gestione dello stato (mantenimento e aggiornamento della validità).
- Processi di revoca (implementazione di meccanismi per revocare le Istanze del Wallet), secondo la Sezione *Revoca dell'Istanza del Wallet* (pagina 312).

In base allo stato dell'Istanza del Wallet e alla richiesta dell'Utente, questo componente interagisce con gli altri componenti dell'Istanza del Wallet.

Componente Issuer

Seguendo la specifica OpenID4VCI e il profilo di implementazione nella Sezione *Emissione di Attestati Elettronici* (pagina 443), questo componente DEVE implementare i protocolli e i flussi di emissione delle Credenziali Elettroniche per richiedere Credenziali Elettroniche ai Credential Issuer.

Componente di Presentazione

Seguendo il profilo di implementazione nella Sezione *Presentazione dell'Attestato Elettronico* (pagina 506), questo componente DEVE essere conforme ai flussi remoti basati su OpenID4VP e ai flussi di prossimità basati su ISO18013-5.

Componente di Backup e Ripristino

Per ogni Credenziale Elettronica emessa all'Istanza del Wallet, questo componente DEVE aggiungere tutti i dati necessari per richiedere l'emissione di quella Credenziale Elettronica durante il ripristino come specificato nella Sezione *Backup e Ripristino* (pagina 320).

Nota

Attualmente la riemissione del PID e dell'IT-Wallet ID non è gestita dal Componente di Backup e Ripristino.

Dashboard e Registro delle Transazioni

Questo componente DEVE fornire le funzionalità di dashboard e di registro delle transazioni della Wallet Unit. In particolare, questo componente DEVE:

- fornire un'interfaccia accessibile all'Utente che consenta all'Utente di accedere alle informazioni di trasparenza delle transazioni;
- mantenere un registro delle transazioni eseguite tramite la Wallet Unit, incluse le transazioni non completate;
- supportare l'interazione dell'Utente con i record di transazione, inclusa la visualizzazione, l'esportazione e la cancellazione.

Archiviazione Sicura

L'Istanza del Wallet DEVE utilizzare questo componente per proteggere gli asset critici e per eseguire in modo sicuro funzioni crittografiche.

10.1.3 Modelli di Interazione della Soluzione Wallet

La Soluzione Wallet supporta questi modelli di interazione:

1. **Utente verso Frontend del Backend del Wallet**: Interazioni basate sul web per la gestione dell'Istanza del Wallet.
2. **Istanza del Wallet verso API del Backend del Wallet**: per la registrazione dell'Istanza del Wallet e l'emissione della Wallet Instance e key Attestation.
3. **Provider di PID/IT-Wallet ID verso API del Backend del Wallet**: Chiamate API sicure per richiedere la revoca dell'Istanza del Wallet.
4. **Utente verso Interfaccia Utente dell'Istanza del Wallet**: per la gestione delle Credenziali Elettroniche (emissione, presentazione, backup, ripristino, eliminazione).
5. **Istanza del Wallet verso Relying Party**: per la presentazione delle Credenziali Elettroniche.

10.1.4 Istanza del Wallet

L'Istanza del Wallet stabilisce un meccanismo forte e affidabile per garantire all'Utente transazioni digitali in modo sicuro e rispettoso della privacy.

L'Istanza del Wallet stabilisce un rapporto di fiducia con i Fornitori di PID e (Q)EAA presentando le Wallet Instance Attestation e Key Attestation durante le interazioni. Queste attestazioni verificabili, fornite dal Fornitore di Wallet, servono ad autenticare la stessa Istanza del Wallet, garantendo l'affidabilità del WSCD, verificando che l'Istanza del Wallet non sia stata revocata e assicurandone l'affidabilità nelle interazioni con gli altri attori dell'ecosistema.

Ciclo di vita dell'Istanza del Wallet

Il Fornitore di Wallet è responsabile dell'implementazione e della fornitura delle Istanze del Wallet, gestendo anche l'intero ciclo di vita.

In questa sezione, vengono presentate macchine a stati per spiegare gli stati delle Istanze del Wallet e delle Credenziali Elettroniche e le loro transizioni e relazioni.

Nota

L'Attestato Elettronico di Dati di Identificazione Personale è un tipo specializzato di Credenziale Elettronica che ha impatti sul ciclo di vita dell'Istanza del Wallet. La revoca dell'Attestato Elettronico di Dati di Identificazione Personale PUÒ avere anche potenziali impatti sugli Attestati Elettronici di Attributi (Qualificati), se questi sono stati emessi utilizzando la presentazione dell'Attestato Elettronico di Dati di Identificazione Personale. Quando non è necessaria la distinzione tra Attestato Elettronico di Dati di Identificazione Personale e Attestati Elettronici di Attributi (Qualificati), viene utilizzato il termine Credenziale Elettronica.

Come mostrato in Fig. 10.2, l'Istanza del Wallet ha quattro stati distinti: **Installato**, **Operativo**, **Valido** e **Disinstallato**. Ogni stato rappresenta uno stato funzionale specifico e determina le azioni che possono essere eseguite.

Nota

Il Fornitore di Wallet DEVE garantire la sicurezza e l'affidabilità delle Istanze del Wallet. Per raggiungere questo obiettivo, il Fornitore di Wallet DEVE verificare periodicamente lo stato di sicurezza e conformità delle Istanze del Wallet.

Transizione a Installato

La macchina a stati inizia con la transizione di installazione dell'Istanza del Wallet (**WI INST**), dove gli Utenti scaricano e installano un'Istanza del Wallet fornita dal Fornitore di Wallet utilizzando lo store ufficiale del sistema operativo del loro dispositivo (questo garantisce l'autenticità tramite controlli di sistema), portando allo stato **Installato**.

Quando lo stato è **Installato**, l'Istanza del Wallet DEVE interagire solo con il Fornitore di Wallet per essere attivata. Quando avviene la revoca dell'Istanza del Wallet, l'Istanza del Wallet DEVE tornare da **Operativo** o **Valido** a **Installato**. La revoca segna la Cryptographic Hardware Key del Wallet, registrata durante l'attivazione (vedi *Transizione a Operativo* (pagina 301)), come non più utilizzabile. La revoca può avvenire nei seguenti casi:

- per motivi tecnici di sicurezza (ad esempio, relativi alla compromissione del materiale crittografico);
- in caso di richieste esplicite dell'Utente (ad esempio, a causa di perdita o furto dell'Istanza del Wallet);
- morte dell'Utente;
- attività illegali segnalate da Organi Giudiziari o di Vigilanza.

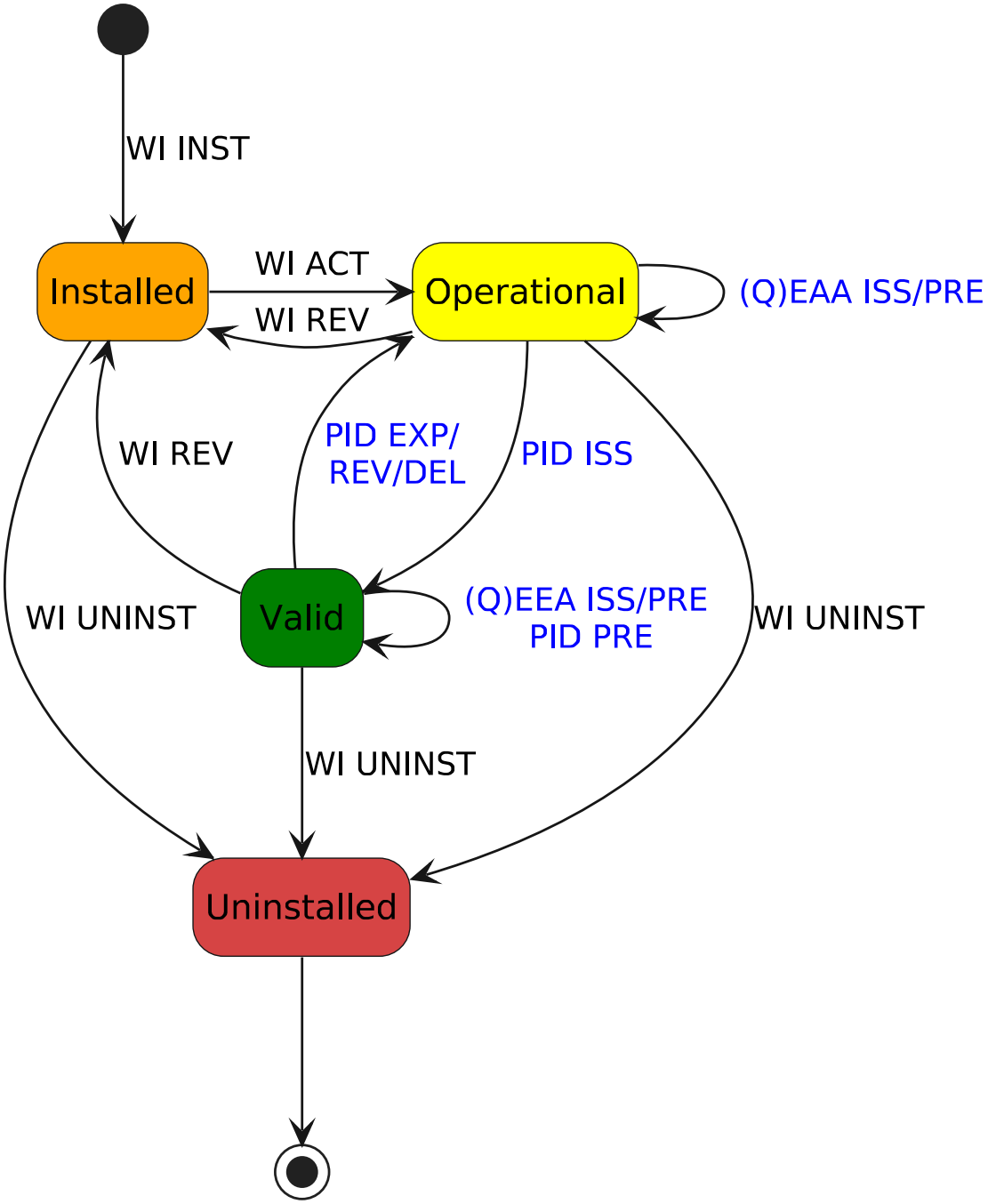


Fig. 10.2: Stati dell'Istanza del Wallet.

Nota

Come indicato nell'ARF (vedi Topic 9 e Topic 38 nell'Allegato 2), la revoca dell'Istanza del Wallet viene effettuata revocando la Wallet Instance Attestation tramite il meccanismo di gestione dello stato. Il Fornitore di Wallet può semplicemente modificare lo stato della Wallet Instance nella Status List di Wallet Instance Attestation, contrassegnandola come revocata, e informare gli Utenti riguardo al motivo della revoca dell'Istanza del Wallet.

Transizione a Operativo

Dopo l'installazione, l'Utente apre l'Istanza del Wallet e inizia un'attivazione (**WI ACT**). In questa fase, un account Utente DEVE essere creato con il Fornitore di Wallet e associato all'Istanza del Wallet tramite il Cryptographic Hardware Key Tag, previa acquisizione del consenso dell'Utente (vedi *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304) per maggiori dettagli, con i test in *WP_024* (pagina 781) e *WP_138* (pagina 814)). Questa associazione consente all'utente di richiedere direttamente al Fornitore di Wallet la revoca dell'Istanza del Wallet.

Nota

Come risultato della creazione dell'account Utente, DEVE essere impostato un meccanismo di autenticazione per l'Utente per interagire con il portale del Fornitore di Wallet. Questa specifica impone l'uso di almeno un secondo fattore per l'autenticazione dell'Utente.

Come parte del processo di attivazione, il Fornitore di Wallet DEVE valutare il sistema operativo, le capacità tecniche generali del dispositivo e l'affidabilità del WSCD, al fine di verificare la conformità ai requisiti tecnici e di sicurezza, l'autenticità e l'integrità dell'Istanza del Wallet installata, e assicurarsi che le chiavi utilizzate per il binding crittografico risiedano in un WSCD sicuro. Dopo la verifica con successo, il Fornitore di Wallet DEVE emettere almeno una Wallet Attestation valido all'Istanza del Wallet, quindi l'Istanza del Wallet entra nello stato **Operativo**.

Inoltre, se non è già stato fatto, gli Utenti DEVONO impostare il loro metodo preferito per sbloccare la loro Istanza del Wallet; questo PUÒ essere realizzato inserendo un PIN o utilizzando l'autenticazione biometrica, come l'impronta digitale o il riconoscimento facciale, secondo le preferenze personali e le capacità del dispositivo (*WP_025* (pagina 781)). Si prega di fare riferimento a *Emissione della Wallet Instance Attestation* (pagina 305).

Nello stato **Operativo**, gli Utenti possono richiedere l'emissione del PID (**PID ISS**), dell'**IT-Wallet ID** (che **non** causa la transizione allo stato **Valido**), o di Attestati Elettronici di Attributi (Qualificati) se il PID non è richiesto nell'emissione (**(Q)EEA ISS**). Inoltre, se le

Credenziali Elettroniche sono Attestati Elettronici di Attributi (Qualificati) e per la presentazione non richiedono il PID, possono essere presentate senza far transitare l'Istanza del Wallet a un altro stato (transizione **(Q)EEA PRE**). L'IT-Wallet ID PUÒ altresì essere presentato mentre l'Istanza del Wallet rimane **Operativa**.

Un'Istanza del Wallet **Valida** DEVE transitare nuovamente allo stato **Operativo** a causa della transizione **PID EXP/REV/DEL**, quando il PID associato scade, viene revocato dal suo Fornitore o viene eliminato dall'Utente.

Transizione a Valido

Una transizione allo stato Valido si verifica solo quando l'Istanza del Wallet ottiene un PID valido (**PID ISS**). L'ottenimento di un IT-Wallet ID NON DEVE causare questa transizione. Nello stato **Valido**, gli Utenti possono ottenere e presentare nuovi Attestati Elettronici di Attributi (Qualificati) (**(Q)EAA ISS/PRE**), e presentare il PID (**PID PRE**). Si prega di fare riferimento a *Emissione di Attestati Elettronici* (pagina 443) e *Presentazione dell'Attestato Elettronico* (pagina 506).

Nota

Gli Utenti possono avere solo un'Istanza del Wallet in stato **Valido** per la stessa Soluzione Wallet. Pertanto, quando un Utente installa e ottiene un PID su una nuova Istanza del Wallet della stessa Soluzione Wallet dallo stesso Fornitore di Wallet, il PID nella precedente Istanza del Wallet DEVE essere revocato e l'Istanza del Wallet diventa **Operativa**.

Transizione a Disinstallato

In tutti gli stati, **Installato**, **Attivato**, **Operativo** o **Valido**, l'Istanza del Wallet può essere rimossa completamente attraverso la transizione di disinstallazione dell'Istanza del Wallet (**WI UNINST**), portando allo stato **Disinstallato**. Se un'Istanza del Wallet è **Disinstallata** termina il suo ciclo di vita (*WP_139* (pagina 814)).

Gestione del Ciclo di Vita dell'Istanza del Wallet

Mentre Fig. 10.2 mostra i diversi stati che un'Istanza del Wallet può acquisire durante il suo ciclo di vita, Fig. 10.3 mostra il punto di vista delle Istanze del Wallet e dei Fornitori di Wallet nella gestione del ciclo di vita dell'Istanza del Wallet e l'effetto sul loro storage locale.

Attraverso un'Istanza del Wallet in stato **Installato**, un Utente è in grado di avviare l'**Attivazione dell'Istanza del Wallet** (**WI ACT**). Come risultato, l'Istanza del Wallet DEVE creare una coppia di Cryptographic Hardware Key. Inoltre, se non è già stato fatto,

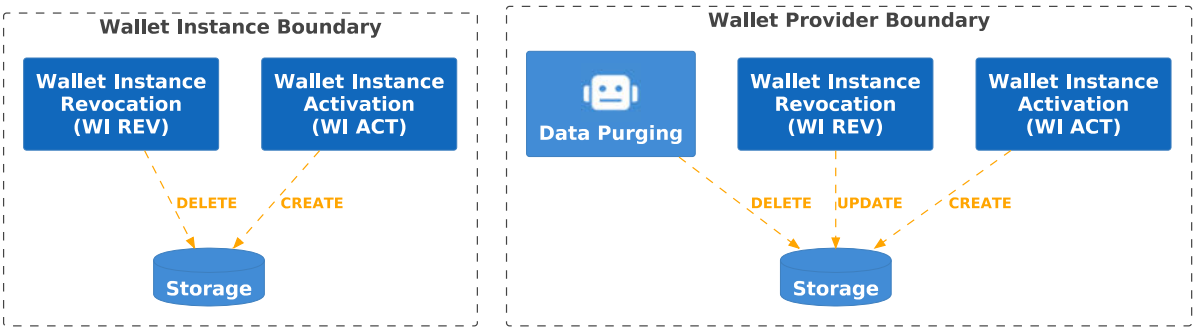


Fig. 10.3: Gestione del Ciclo di Vita dell'Istanza del Wallet.

gli Utenti DEVONO impostare il loro metodo preferito per sbloccare la loro Istanza del Wallet. Come risultato della **Revoca dell'Istanza del Wallet** (**WI REV**), l'Istanza del Wallet DEVE eliminare le coppie di Cryptographic Hardware Key (*WP_149* (pagina 814)).

Un Fornitore di Wallet invece è responsabile per:

- **Attivazione dell'Istanza del Wallet** (**WI ACT**): un account Utente DEVE essere creato e associato all'Istanza del Wallet tramite il Cryptographic Hardware Key Tag. Come risultato della creazione dell'account Utente, DEVE essere impostato un meccanismo di autenticazione di almeno due fattori per l'Utente per interagire con il portale del Fornitore di Wallet.
- **Revoca dell'Istanza del Wallet** (**WI REV**): Per motivi di sicurezza tecnica o a seguito dell'intervento di entità esterne (ad esempio Utenti o Autorità di Vigilanza), l'istanza del Wallet viene revocata, con conseguente revoca della Wallet Instance Attestation, che viene riportata nella Wallet Instance Attestation Status List. Inoltre, il Wallet Cryptographic Hardware Key Tag DEVE essere eliminato dall'account dell'utente.
- **Eliminazione dei Dati**: attraverso una richiesta esplicita degli Utenti, l'account Utente presso il Fornitore di Wallet DEVE essere rimosso dallo storage locale.

Funzionalità dell'Istanza del Wallet

Un'Istanza del Wallet DEVE supportare le seguenti funzionalità:

- Registrazione del Wallet (dettagliata in *Inizializzazione e Registrazione dell'Istanza del Wallet* (pagina 304)),
- Emissione della Wallet Instance Attestation (descritta in dettaglio in *Emissione della Wallet Instance Attestation* (pagina 305)),
- Emissione della Key Attestation (descritta in dettaglio in *Emissione della Key Attestation* (pagina 308)),
- Revoca del Wallet (dettagliata in *Revoca dell'Istanza del Wallet* (pagina 312)) e

- Cancellazione degli attributi presentati (dettagliata in *Eliminazione degli Attributi dell'Utente* (pagina 316)).

Ciascuna funzionalità è descritta in dettaglio nelle sezioni seguenti.

Nota

I dettagli forniti di seguito sono non normativi e hanno lo scopo di chiarire le funzionalità della Registrazione dell'Istanza del Wallet. L'implementazione effettiva può variare in base al caso d'uso specifico e ai requisiti del Fornitore di Wallet.

Inizializzazione e Registrazione dell'Istanza del Wallet

Questo processo consente all'Utente che ha appena installato l'applicazione di registrare l'Istanza del Wallet presso il Backend del Fornitore di Wallet. Durante questo processo, l'applicazione richiederà un'attestazione di sicurezza e integrità dal produttore del sistema operativo, che associa anche una coppia di chiavi a lunga durata memorizzata in un adeguato storage sicuro all'interno del dispositivo stesso. Questa attestazione sarà convalidata dal Fornitore di Wallet e, se la convalida ha esito positivo, il Fornitore di Wallet autenticherà l'Istanza del Wallet. Per i dettagli vedere *Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 705).

Avvertimento

Durante la fase di registrazione dell'Istanza del Wallet presso il Fornitore di Wallet è necessario anche associare l'Istanza del Wallet ad un Utente specifico, autenticando l'Utente con il Fornitore di Wallet. Il meccanismo di autenticazione è a discrezione del Fornitore di Wallet e non sarà trattato all'interno di queste linee guida, poiché ogni Fornitore di Wallet potrebbe avere già implementato i propri sistemi di autenticazione dell'Utente.

Nota

Il Fornitore di Wallet DOVREBBE associare l'Istanza del Wallet (tramite l'identificatore `hardware_key_tag`) a un Utente specifico identificato in modo univoco all'interno dei sistemi del Fornitore di Wallet. Questo sarà utile per il ciclo di vita dell'Istanza del Wallet e per una futura revoca. Per i dettagli far riferimento all'*Istanza dell'Applicazione Mobile* (pagina 705).

Emissione della Wallet Instance Attestation

Questa sezione descrive come il Fornitore di Wallet emette una Wallet Instance Attestation.

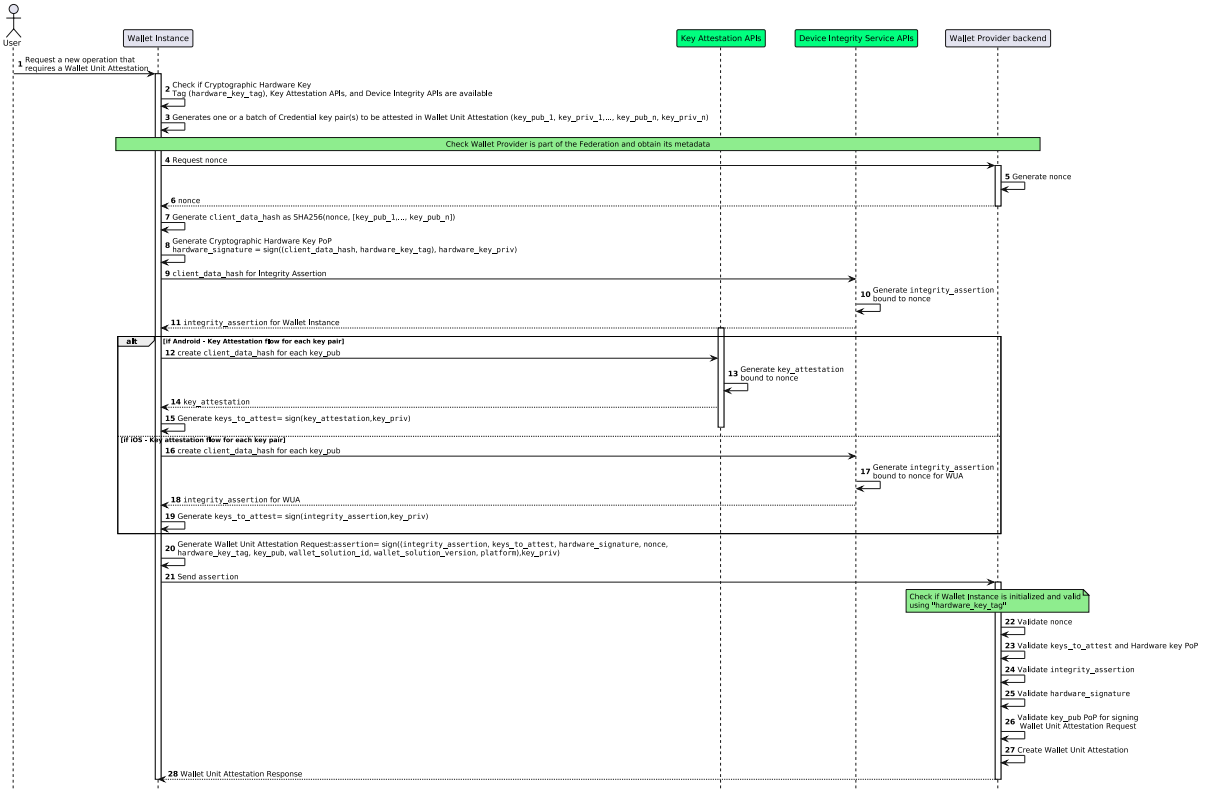


Fig. 10.4: Diagramma di Sequenza per l'acquisizione della Wallet Instance Attestation.

Passo 1: L'Utente avvia una nuova operazione che richiede l'acquisizione di una Wallet Instance Attestation.

Passi 2-3: L'Istanza del Wallet DEVE:

1. Verificare l'esistenza delle Cryptographic Hardware Keys. Se non esistono, è necessaria la reinizializzazione dell'Istanza del Wallet (*WP_140a* (pagina 814)).
2. Generare una coppia di chiavi asimmetriche effimere per la Wallet Instance Attestation (`ephemeral_key_pub`, `ephemeral_key_priv`), collegando la chiave pubblica all'attestazione (*WP_026* (pagina 781)).
3. Verificare l'appartenenza del Fornitore di Wallet alla federazione e recuperare i suoi metadati (*WP_023* (pagina 781)).

Passi 4-6 (Recupero del Nonce): L'Istanza del Wallet richiede un `nonce` all'endpoint *Endpoint Nonce della Soluzione Wallet* (pagina 572) del Backend del Fornitore del Wallet (*WP_140b* (pagina 814)). Il `nonce` deve essere imprevedibile e funge da principale difesa contro gli attacchi di replay.

Il `nonce` DEVE garantire un utilizzo singolo entro un intervallo di tempo predeterminato.

In caso di richiesta riuscita, il Fornitore di Wallet genera e restituisce il valore del nonce all'Istanza del Wallet.

Passo 7: L'Istanza del Wallet esegue le seguenti azioni (*WP_140c* (pagina 814)):

- Crea `client_data`, un oggetto JSON che include il `nonce` e la thumbprint JWK di `ephemeral_key_pub`.
- Calcola i corrispondenti hash, `client_data_hash` applicando l'algoritmo SHA256 a `client_data`.

Di seguito è riportato un esempio non normativo dell'oggetto JSON `client_data`.

```
{
  "nonce": "i4ThI2Jhbu81i8mqyWEuDG5t",
  "jwk_thumbprint": "vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"
}
```

Passi 8-10: L'Istanza del Wallet:

- produce un valore `hardware_signature` firmando `client_data_hash` con la chiave privata dell'Hardware del Wallet, servendo come prova di possesso delle Cryptographic Hardware Keys (*WP_140d* (pagina 814)).
- richiede al Servizio di Integrità del Dispositivo di creare un valore `integrity_assertion` collegato al `client_data_hash`.
- riceve un valore `integrity_assertion` firmato per l'Istanza del Wallet dal Servizio di Integrità del Dispositivo, autenticato dall'OEM (*WP_140e* (pagina 814)).

Nota

`integrity_assertion` è un payload personalizzato generato dal Servizio di Integrità del Dispositivo, firmato dall'OEM del dispositivo e codificato in base64 per avere uniformità tra diversi dispositivi.

Passi 11-12 (Richiesta di Emissione della Wallet Instance Attestation): L'Istanza del Wallet:

- Costruisce la Wallet Instance Attestation Request sotto forma di JWT. Questo JWT include l'`integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, `platform`, `wallet_solution_id`, `wallet_solution_version` e altri parametri relativi alla configurazione (vedi *Tabella del Corpo della Richiesta di Wallet Instance Attestation* (pagina 577)) ed è firmato utilizzando la chiave privata la cui chiave pubblica è indicata nella richiesta tramite `cnf` (*WP_140–141* (pagina 814)).

- Invia la Wallet Instance Attestation Request all'endpoint *Endpoint di Emissione della Wallet Instance Attestation* (pagina 576) del Backend del Fornitore del Wallet.

L'Istanza del Wallet DEVE inviare il JWT firmato della Richiesta di Wallet Instance Attestation come parametro `assertion` nel corpo di una richiesta HTTP all'endpoint *Endpoint di Emissione della Wallet Instance Attestation* (pagina 576) del Fornitore di Wallet (*WP_142* (pagina 814)).

Passi 13-17: Il Backend del Fornitore del Wallet valuta la Richiesta di Wallet Instance Attestation e DEVE eseguire i seguenti controlli (*WP_143* (pagina 814)):

1. La richiesta DEVE includere tutti i parametri dell'intestazione HTTP richiesti come definito in *Richiesta di Emissione della Wallet Instance Attestation* (pagina 576) (*WP_143a* (pagina 814)).
2. La firma della Richiesta di Wallet Instance Attestation DEVE essere valida e verificabile utilizzando la `jwk` fornita (*WP_143b* (pagina 814)).
3. Il valore `nonce` DEVE essere stato generato dal Fornitore di Wallet e non essere stato utilizzato in precedenza (*WP_143c* (pagina 814)).
4. DEVE esistere un'Istanza del Wallet valida e attualmente registrata associata al `hardware_key_tag` (*WP_143d* (pagina 814)).
5. Il `client_data` DEVE essere ricostruito utilizzando il `nonce` e la chiave pubblica `jwk`. Il valore del parametro `hardware_signature` viene quindi convalidato utilizzando la chiave pubblica della Cryptographic Hardware Key registrata associata all'Istanza del Wallet (*WP_143e* (pagina 814)).
6. L'`integrity_assertion` DEVE essere convalidato secondo le linee guida del produttore del dispositivo. I controlli specifici eseguiti dal Fornitore di Wallet sono dettagliati nella documentazione del produttore del sistema operativo (*WP_143f* (pagina 814)).
7. Il dispositivo in uso DEVE essere privo di difetti di sicurezza noti e soddisfare i requisiti minimi di sicurezza definiti dal Fornitore di Wallet.
8. L'URL nel parametro `iss` DEVE corrispondere all'identificatore URL del Fornitore di Wallet (*WP_143g* (pagina 814)).

Al completamento con successo di tutte le verifiche, il Fornitore di Wallet emette una Wallet Instance Attestation che soddisfa *la durata della Wallet Instance Attestation* (pagina 290) (*WP_028* (pagina 781), *WP_144* (pagina 814)).

Passo 18 (Risposta di Emissione della Wallet Instance Attestation): Al completamento con successo, il Fornitore di Wallet DEVE restituire una risposta di conferma utilizzando il codice di stato 200 e il Content-Type `application/json`, contenente la Wallet Instance Attestations firmate dal Fornitore di Wallet. Il Fornitore di Wallet DEVE restituire la Wallet Instance Attestation in formato `JWT`. L'Istanza del Wallet eseguirà quindi la verifica di sicurezza e integrità

della Wallet Instance Attestations ricevute, oltre alla verifica di fiducia del relativo emittente (*WP_030–031* (pagina 781)).

Di seguito è riportato un esempio non normativo della risposta.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "wallet_instance_attestation": "omppc3N1ZXJBdXRohEOhASaiBE...dElEAnFlbGVtZW50SWRl"
}
```

Emissione della Key Attestation

Questa sezione descrive come il Fornitore di Wallet emette una Key Attestation.

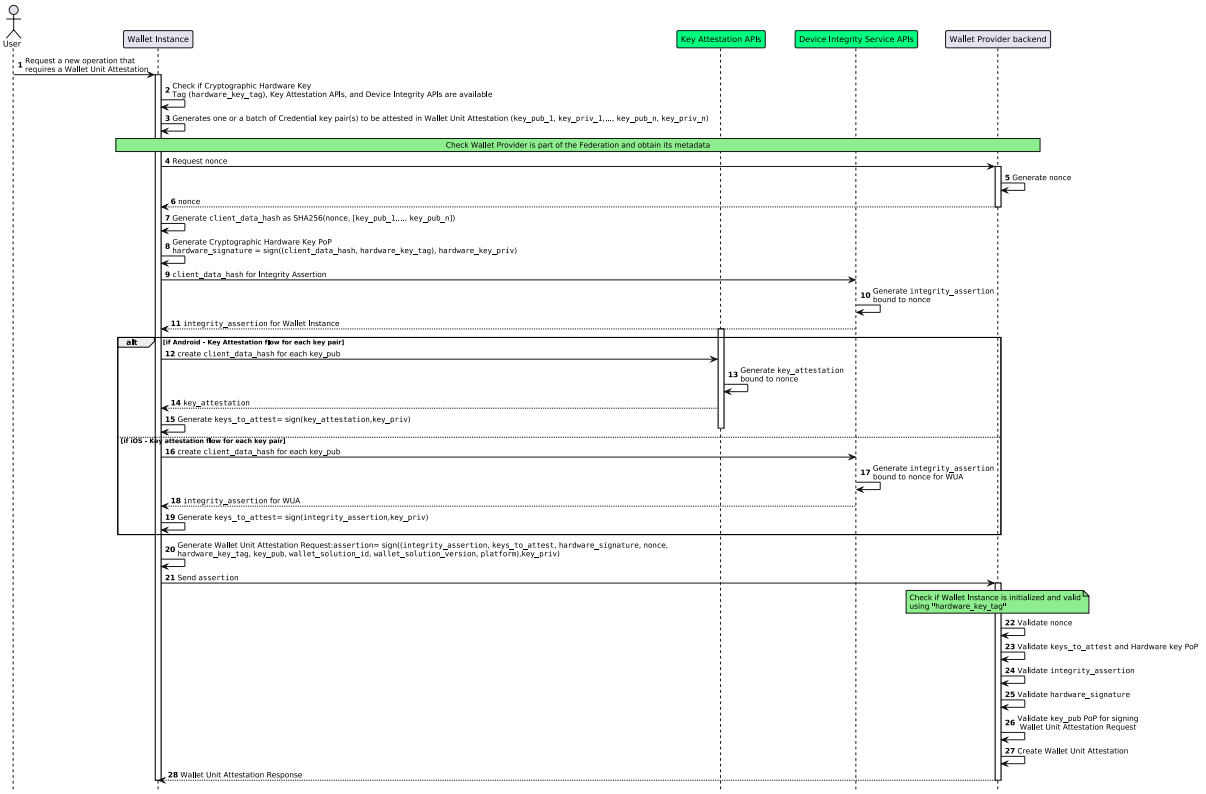


Fig. 10.5: Diagramma di Sequenza per l'acquisizione della Key Attestation.

Passo 1: L'Utente avvia una nuova operazione che richiede l'acquisizione di una Key Attestation.

Passi 2-3: L'Istanza del Wallet DEVE:

1. Verificare l'esistenza delle Cryptographic Hardware Keys. Se non esistono, è necessaria la reinizializzazione dell'Istanza del Wallet (*WP_140a* (pagina 814)).

2. Generare una o più coppie di chiavi asimmetriche delle Credenziali da attestare nella Key Attestation (`key_pub_1`, `key_priv_1`, ..., `key_pub_n`, `key_priv_n`).
3. Verificare l'appartenenza del Fornitore di Wallet alla federazione e recuperare i suoi metadati (*WP_023* (pagina 781)).

Passi 4-6 (Recupero del Nonce): L'Istanza del Wallet richiede un `nonce` all'endpoint *Endpoint Nonce della Soluzione Wallet* (pagina 572) del Backend del Fornitore del Wallet (*WP_140b* (pagina 814)). Il `nonce` deve essere imprevedibile e funge da principale difesa contro gli attacchi di replay.

Il `nonce` DEVE garantire un utilizzo singolo entro un intervallo di tempo predeterminato.

In caso di richiesta riuscita, il Fornitore di Wallet genera e restituisce il valore del nonce all'Istanza del Wallet.

Passo 7: L'Istanza del Wallet esegue le seguenti azioni (*WP_140c* (pagina 814)):

- Crea `client_data`, un oggetto JSON che include il `nonce` e un campo `jwk_thumbprints` contenente un array JSON delle impronte JWK corrispondenti alle chiavi pubbliche (`key_pub_1`, ..., `key_pub_n`).
- Calcola i corrispondenti hash, `client_data_hash` applicando l'algoritmo SHA256 a `client_data`.

Di seguito è riportato un esempio non normativo dell'oggetto JSON `client_data`.

```
{
  "nonce": "i4ThI2Jhbu81i8mqyWEuDG5t",
  "jwk_thumbprints": ["vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"]
}
```

Passi 8-11: L'Istanza del Wallet:

- produce un valore `hardware_signature` firmando `client_data_hash` con la chiave privata dell'Hardware del Wallet, servendo come prova di possesso delle Cryptographic Hardware Keys (*WP_140d* (pagina 814)).
- richiede al Servizio di Integrità del Dispositivo di creare un valore `integrity_assertion` collegato al `client_data_hash`.
- riceve un valore `integrity_assertion` firmato per l'Istanza del Wallet dal Servizio di Integrità del Dispositivo, autenticato dall'OEM (*WP_140e* (pagina 814)).

Nota

`integrity_assertion` è un payload personalizzato generato dal Servizio di Integrità del Dispositivo, firmato dall'OEM del dispositivo e codificato in base64 per avere uniformità tra diversi dispositivi.

Nota

Nel caso del sistema operativo Android, il flusso prosegue in base con i **Passi 12-15**, mentre per il caso di iOS, il flusso prosegue con i **Passi 16-19**.

Passi 12-15: L'Istanza del Wallet:

- richiede all'API di Key Attestation di creare un valore `key_attestation` collegato per ogni `client_data_hash` per ciascuna `key_pub`.
- riceve un valore `key_attestation` firmato dall'API di Key Attestation, autenticato dall'OEM.
- genera un valore `keys_to_attest` firmando il `key_attestation` utilizzando la chiave privata della coppia di chiavi di credenziale generata inizialmente (`priv_key`).

Passi 16-19: L'Istanza del Wallet:

- richiede al Servizio di Integrità del Dispositivo di creare un valore `integrity_assertion` collegato per ogni `client_data_hash` per ogni `key_pub`.
- riceve un valore `integrity_assertion` firmato per la Key Attestation dal Servizio di Integrità del Dispositivo, autenticato dall'OEM.
- genera un valore `keys_to_attest` firmando l'`integrity_assertion` ottenuto per la key Attestation utilizzando la chiave privata della coppia di chiavi di credenziale generata inizialmente (`priv_key`).

Passi 20-21 (Richiesta di Emissione della Key Attestation): L'Istanza del Wallet:

- Costruisce la Key Attestation Request sotto forma di JWT. Questo JWT include l'`integrity_assertion`, `keys_to_attest`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, `platform`, `wallet_solution_id`, `wallet_solution_version` e altri parametri relativi alla configurazione (vedi *Tabella del Corpo della Richiesta di Key Attestation* (pagina 584)). La Key Attestation Request DEVE essere firmata utilizzando la chiave privata corrispondente alla chiave pubblica inclusa nella richiesta, tramite il parametro `cnf` (primo elemento di `keys_to_attest`) (*WP_140–141* (pagina 814)).
- Invia la Key Attestation Request all'endpoint *Endpoint di Emissione della Key Attestation* (pagina 583) del Backend del Fornitore del Wallet.

Nota

`keys_to_attest` contiene un oggetto `key_attestation` nel caso di Android e un oggetto `integrity_assertion` nel caso di iOS.

L'Istanza del Wallet DEVE inviare il JWT firmato della Richiesta di Key Attestation come parametro `assertion` nel corpo di una richiesta HTTP all'endpoint *Endpoint di Emissione della Key Attestation* (pagina 583) del Fornitore di Wallet (*WP_142* (pagina 814)).

Passi 22-27: Il Backend del Fornitore del Wallet valuta la Richiesta di Key Attestation e DEVE eseguire i seguenti controlli (*WP_143* (pagina 814)):

1. La richiesta DEVE includere tutti i parametri dell'intestazione HTTP richiesti come definito in *Richiesta di Emissione della Key Attestation* (pagina 583) (*WP_143a* (pagina 814)).
2. La firma della Richiesta di Key Attestation DEVE essere valida e verificabile utilizzando la `jwk` fornita (*WP_143b* (pagina 814)).
3. Il valore `nonce` DEVE essere stato generato dal Fornitore di Wallet e non essere stato utilizzato in precedenza (*WP_143c* (pagina 814)).
4. DEVE esistere un'Istanza del Wallet valida e attualmente registrata associata al `hardware_key_tag` (*WP_143d* (pagina 814)).
5. La firma del parametro `keys_to_attest` deve essere prima validata utilizzando la `jwk` fornita, e il suo valore (`key_attestation` nel caso di Android o `integrity_assertion` nel caso di iOS) DEVE essere validato secondo le linee guida del produttore del dispositivo.
6. Il `client_data` DEVE essere ricostruito utilizzando il `nonce` e le rispettive thumbprint JWKs `[key_pub_1,...,key_pub_n]`. Il valore del parametro `hardware_signature` viene quindi convalidato utilizzando la chiave pubblica della Cryptographic Hardware Key registrata associata all'Istanza del Wallet (*WP_143e* (pagina 814)).
7. L'`integrity_assertion` DEVE essere convalidato secondo le linee guida del produttore del dispositivo. I controlli specifici eseguiti dal Fornitore di Wallet sono dettagliati nella documentazione del produttore del sistema operativo (*WP_143f* (pagina 814)).
8. Il dispositivo in uso DEVE essere privo di difetti di sicurezza noti e soddisfare i requisiti minimi di sicurezza definiti dal Fornitore di Wallet.
9. L'URL nel parametro `iss` DEVE corrispondere all'identificatore URL del Fornitore di Wallet (*WP_143g* (pagina 814)).

Al completamento con successo di tutte le verifiche, il Fornitore di Wallet emette una Key Attestation valida per almeno un mese (*WP_144* (pagina 814)).

Passo 28 (Risposta di Emissione della Key Attestation): Al completamento con successo, il Fornitore di Wallet DEVE restituire una risposta di conferma con codice di stato `200` e Content-Type `application/json`. La risposta DEVE contenere le Key Attestations firmate dal Fornitore di Wallet in formato JWT. L'Istanza del Wallet DEVE eseguire la verifica di sicurezza e integrità delle Key Attestations ricevute, insieme alla verifica di fiducia già valutata sull'Emittente (*WP_030–031* (pagina 781)).

Di seguito è riportato un esempio non normativo della risposta.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "key_attestation": "omppc3N1ZXJBdXRohEOhASaiBE...dElEAnFlbGVtZW50SWRl"
}
```

Revoca dell'Istanza del Wallet

Questa sezione descrive le entità coinvolte e le modalità per richiedere la revoca di un'Istanza del Wallet.

Il Fornitore di Wallet DEVE garantire la sicurezza e l'affidabilità delle Istanze del Wallet, mantenendole aggiornate e conformi ai requisiti di sicurezza. Quando, per motivi tecnici di sicurezza (ad esempio, relativi alla compromissione del materiale crittografico) la sicurezza dell'Istanza del Wallet è compromessa, il Fornitore di Wallet DEVE revocare l'Istanza del Wallet.

Come mostrato in Fig. 10.6, altri attori POSSONO attivare il processo di revoca dell'Istanza del Wallet (*WP_007–009* (pagina 778)):

- **Utenti**, collegandosi al portale web del Fornitore di Wallet dalla propria Istanza del Wallet o utilizzando un browser esterno.
- **PID Provider e EAA Provider di IT-Wallet ID** quando notificati dalla Fonte Autentica del PID/IT-Wallet ID (ANPR) del decesso dell'Utente.
- **Autorità Legali o Organismo di Supervisione** in casi di comprovate attività illegali.

Indipendentemente da chi ha attivato il processo di revoca, il Wallet Provider DEVE implementare un meccanismo per informare gli Utenti quando la loro Wallet Unit viene revocata (*WP_034* (pagina 781)):

- Inviando un avviso presso un recapito verificato (ad esempio recapito e-mail o telefonico) e se possibile anche presso l'Istanza del Wallet revocata.
- Inviando la notifica entro massimo 24 ore.
- Includendo una spiegazione chiara e semplice dei motivi della revoca, con eventuali istruzioni per riattivare l'Istanza del Wallet.

Nota

- Il flusso per la Revoca dell'Istanza del Wallet attivato dall'Utente è dettagliato di seguito.

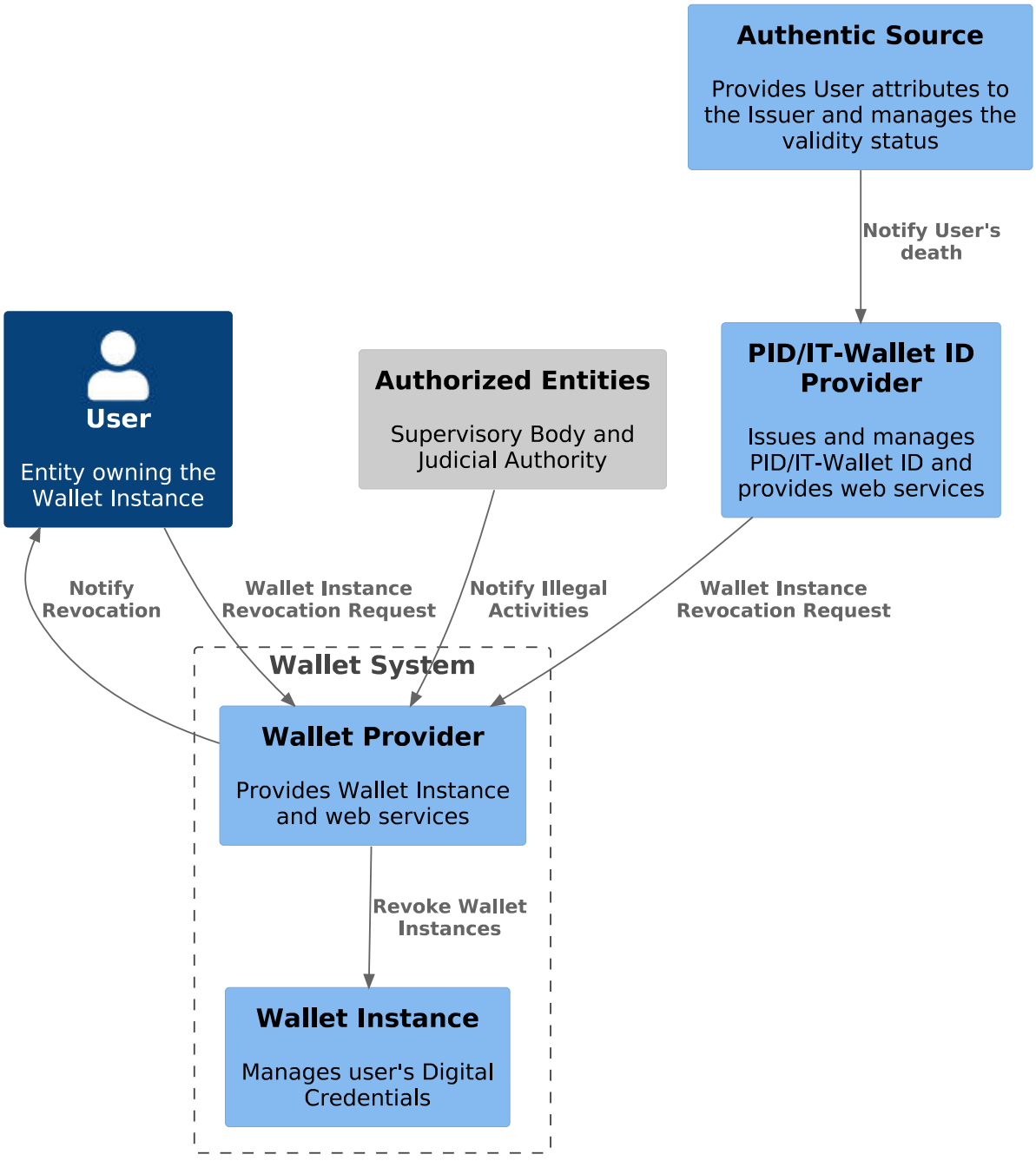


Fig. 10.6: Entità coinvolte nel processo di revoca dell'Istanza del Wallet.

- L'endpoint utilizzato dal Provider di PID/IT-Wallet ID è dettagliato nel Catalogo e-Service del Fornitore di Wallet PDND (vedere la Sezione *Catalogo e-Service PDND del Fornitore di Wallet* (pagina 591) per i dettagli tecnici).
- Il flusso per le Entità Autorizzate (ad esempio, Organismi di Supervisione) è fuori dall'ambito di questa specifica, sarà gestito da ciascun Fornitore di Wallet.

Richiesta di Revoca dell'Istanza del Wallet

Gli Utenti POSSONO richiedere la revoca dell'Istanza del Wallet (*WP_032* (pagina 781)):

- *Selezionando la funzionalità di revoca all'interno dalla propria Istanza del Wallet*: questa funzionalità può essere utilizzata dagli Utenti prima di cambiare il proprio telefono.
- *Utilizzando un user agent esterno*: questo copre i casi in cui gli Utenti perdono il loro dispositivo e quindi l'accesso alla loro Istanza del Wallet.

In entrambi i casi, utilizzando il portale del Fornitore di Wallet (*WP_005–006* (pagina 778)):

- Gli Utenti DEVONO autenticarsi con almeno un meccanismo di autenticazione a due fattori, o avere una sessione attiva che soddisfi questo requisito.
- Il Fornitore di Wallet DEVE consentire agli Utenti di visualizzare lo stato delle loro Istanze del Wallet associate alla loro sessione autenticata e chiedere la revoca, inviando una Richiesta di Recupero o Revoca dell'Istanza del Wallet (*WP_033* (pagina 781), e *WP_145* (pagina 814)), a seconda dei casi, all'endpoint *Endpoint di Gestione dell'Istanza di Wallet* (pagina 572) del Backend del Fornitore del Wallet.

Di seguito è riportato un esempio non normativo di una Richiesta di Recupero delle Istanze del Wallet.

```
GET /wallet-instances HTTP/1.1
Host: walletprovider.example.com
```

In caso di recupero riuscito, il Fornitore di Wallet DEVE restituire una risposta di conferma, con lo stato di tutte le Istanze del Wallet associate all'Utente (*WP_146* (pagina 814)). Di seguito è riportato un esempio non normativo di una Risposta di Recupero delle Istanze del Wallet.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

[
  {
    "id": "f7b2a8d9",
    "status": "ACTIVE",
    "issued_at": "2024-03-12T10:00:00Z"
```

```
  },
  {
    "id": "g8b235c4",
    "status": "REVOKED",
    "issued_at": "2024-02-28T15:30:00Z"
  }
]
```

Una volta che l'Utente identifica l'Istanza del Wallet da revocare, può essere inviata una Richiesta di Revoca dell'Istanza del Wallet all'endpoint, includendo l'ID dell'Istanza del Wallet come parametro di percorso (*WP_147* (pagina 814)). Di seguito è riportato un esempio non normativo di una Richiesta di Revoca dell'Istanza del Wallet (*WP_148* (pagina 814)).

```
PATCH /wallet-instances/{f7b2a8d9} HTTP/1.1
Host: wallet-provider.example.org
Content-Type: application/json

{
  "status": "REVOKED"
}
```

Risposta alla Revoca dell'Istanza del Wallet

In caso di revoca riuscita, il Fornitore di Wallet DEVE restituire una risposta di conferma. Di seguito è riportato un esempio non normativo di una Risposta di Revoca dell'Istanza del Wallet.

```
HTTP/1.1 204 No Content
```

Meccanismi di Verifica della Revoca

La verifica della validità dell'Istanza del Wallet DEVE essere eseguita:

- **Durante l'emissione dell'Attestato Elettronico di Attributi** da parte dei Fornitori di Attestati. Solo le istanze del Wallet in stato Operativo o Valido possiedono una Wallet Instance Attestation e una Key Attestation valide. La verifica della validità di un'Istanza del Wallet viene eseguita indirettamente dai Fornitori di Attestati tramite il controllo della presenza di una Wallet Instance Attestation valida (ossia non scaduta, non revocata — verificando la Wallet Instance Attestation Status List — e firmata da un Fornitore di Portafoglio fidato).
- **Durante il periodo di validità dell'Attestato Elettronico di Attributi** da parte dei Fornitori di Attestati, ogni 24 ore mediante la verifica della Wallet Instance Attestation Status List del dell'istanza del Wallet che detiene il PID e la lista di stato della Key Attestation. Infatti, se la Wallet Instance Attestation o Key Attestation viene revocata,

il PID in essa contenuto DEVE essere revocato. Di conseguenza, qualsiasi altro Attestato Elettronico ottenuto attraverso la presentazione del PID DEVE anch'esso essere revocato.

- **Durante il ciclo di vita dell'Istanza del Wallet** da parte dell'Istanza del Wallet. Ogni Fornitore di Wallet può implementare metodi diversi per consentire a un'Istanza del Wallet di verificarne il suo stato. La specifica IT Wallet RACCOMANDA di utilizzare la Status List della Wallet Instance Attestation.

Nota

Nell'attuale versione della specifica, i Fornitori di Attestati vengono notificati direttamente della revoca di un'Istanza del Wallet dal Fornitore di Wallet tramite un servizio e-service PDND.

Nota

Durante la fase di presentazione degli Attestati Elettronici, una Relying Party può verificare indirettamente la revoca dell'Istanza del Wallet verificando la revoca degli Attestati Elettronici. Quando un'Istanza del Wallet viene revocata, il Fornitore di Wallet revoca le corrispondenti Wallet Instance Attestation, innescando la revoca degli Attestati Elettronici da parte del Fornitore di Attestati Elettronici.

Eliminazione degli Attributi dell'Utente

Questa funzionalità dell'Istanza del Wallet consente agli Utenti di ottenere un elenco di tutte le Relying Party a cui sono stati presentati Attestati Elettronici. Successivamente, gli Utenti possono richiedere l'eliminazione degli attributi di uno o più Attestati Elettronici presentati a una Relying Party di loro scelta. La Relying Party DEVE individuare univocamente tali Attestati Elettronici applicando l'identity matching come specificato in *Identity Matching e Identity Reconciliation* (pagina 567). Di seguito viene presentato il flusso di alto livello relativo a questa interazione (*WP_115* (pagina 810)).

Passo 1: L'Utente richiede l'eliminazione degli attributi invocando la funzione di eliminazione degli attributi dell'Istanza del Wallet.

Passo 2: L'Istanza del Wallet raccoglie tutti i dati delle transazioni e mostra all'Utente l'elenco delle Relying Party con cui ha avuto interazioni durante il ciclo di vita dell'Istanza del Wallet e che sono in possesso degli attributi dell'Utente (*WP_115a* (pagina 810)).

Passo 3: L'Utente seleziona la Relying Party di destinazione per l'eliminazione degli attributi.

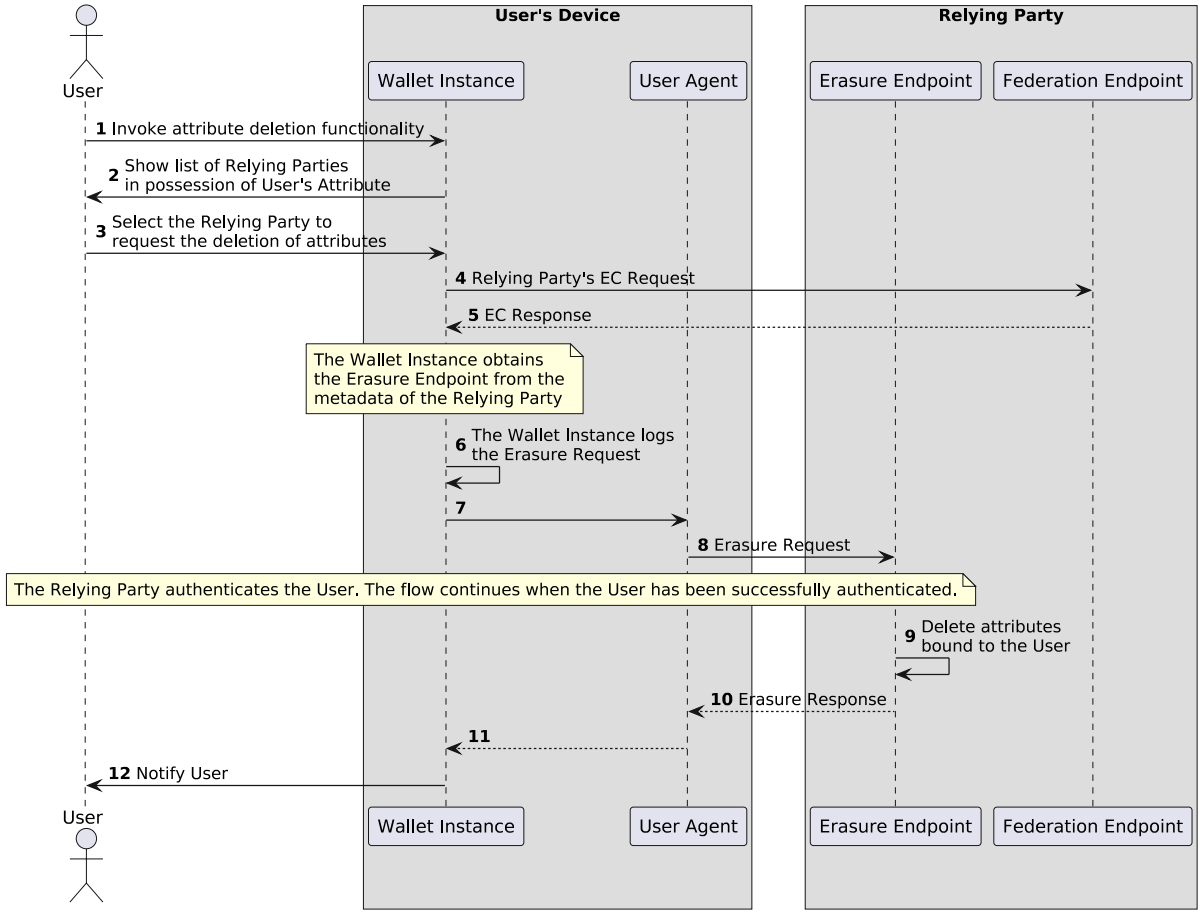


Fig. 10.7: Processo di Eliminazione degli Attributi dell'Utente.

Passi 4 - 5: L'Istanza del Wallet ottiene la Entity Configuration della Relying Party all'endpoint ./well-known/ della Federazione. L'URL o l'Endpoint di Cancellazione (`erasure_endpoint`) può essere trovato all'interno del parametro dei metadati (*WP_116* (pagina 810)).

Passo 6: L'Istanza del Wallet registra le informazioni rilevanti della Richiesta di Cancellazione come definito in *Dashboard dell'Istanza del Wallet e Registrazione delle Transazioni* (pagina 319). Questi log DEVONO includere almeno (*WP_117a* (pagina 810)):

- la data della richiesta,
- la Relying Party a cui è stata fatta la richiesta,
- gli attributi di cui è stata richiesta la rimozione.

Passi 7 - 8: L'Istanza del Wallet reindirizza l'Utente all'Endpoint di Cancellazione. DEVE inoltre garantire che sia presente un meccanismo di callback per consentire all'User-Agent di notificare all'Istanza del Wallet (e quindi all'Utente) dopo la Risposta di Cancellazione (*WP_118* (pagina 810)). I dettagli sulla Richiesta di Cancellazione si trovano in relying-party-provider-backend-endpoint:Endpoint di Cancellazione della Relying Party del Provider Backend (*WP_117* (pagina 810)).

Nota

La pagina web della Relying Party autenticherà l'Utente con un Livello di Garanzia appropriato utilizzando qualsiasi metodo come CIE o la presentazione del PID/IT-Wallet ID. Il meccanismo specifico utilizzato per l'autenticazione è lasciato alla Relying Party. Se l'Utente non si autentica tramite l'Istanza del Wallet, l'identity matching e l'identity reconciliation POSSONO usare uno schema nazionale di autenticazione preesistente, ove possibile, come specificato in *Identity Matching e Identity Reconciliation* (pagina 567). Dopo aver autenticato l'Utente, la Relying Party PUÒ richiedere all'Utente di eseguire ulteriori passaggi necessari per l'eliminazione degli attributi, ad esempio, potrebbe richiedere all'Utente di confermare l'operazione di eliminazione.

Passo 9: Dopo aver autenticato con successo l'Utente, la Relying Party DEVE individuare univocamente uno o più Attestati Elettronici per i quali l'Utente richiede la rimozione applicando l'identity matching, e DEVE eliminare i relativi attributi in suo possesso.

Passo 10: La Relying Party restituisce la Risposta di Cancellazione sotto forma di Risposta HTTP all'User-Agent e include l'URL di callback se fornito nella Richiesta di Cancellazione. I dettagli sulla Risposta di Cancellazione si trovano in relying-party-provider-backend-endpoint: Endpoint di Cancellazione della Relying Party del Provider Backend.

Passi 11 - 12: L'User-Agent utilizza il metodo implementato per restituire la Risposta di Cancellazione all'Istanza del Wallet. Infine, l'Utente viene notificato tramite l'Istanza del Wallet

riguardo all'esito della Risposta di Cancellazione (*WP_119* (pagina 810)).

Dashboard dell'Istanza del Wallet e Registrazione delle Transazioni

L'Istanza del Wallet DEVE mantenere un registro delle transazioni sicuro e locale al Wallet, al fine di garantire trasparenza, tracciabilità e consapevolezza dell'Utente rispetto alle azioni eseguite tramite l'Istanza del Wallet, come definito in EIDAS-ARF (Annex 2.02 HLRs, Topic 19). Il registro DEVE coprire tutte le transazioni (incluse quelle non completate) nelle seguenti categorie:

- rilascio e ri-rilascio delle credenziali;
- presentazione delle credenziali alle Relying Party;
- richieste di cancellazione dei dati inviate alle Relying Party.

Per ciascuna di tali transazioni, l'Istanza del Wallet DEVE creare un record di transazione corrispondente contenente gli elementi informativi previsti dai requisiti ARF applicabili per la relativa categoria di registrazione, e DEVE aggiornarlo con l'esito finale, ove applicabile.

Per le transazioni di presentazione, i valori degli attributi NON DEVONO essere registrati. Qualora una richiesta di presentazione contenga dati transazionali, l'Istanza del Wallet DEVE registrare il valore di tali dati transazionali esclusivamente nella misura esplicitamente richiesta da EUDI-TS 12 in relazione all'attestazione richiesta e nel rispetto dei principi di minimizzazione dei dati. In assenza di tale requisito, i valori dei dati transazionali NON DEVONO essere registrati.

Per supportare l'accesso dell'Utente al registro delle transazioni, l'Istanza del Wallet DEVE fornire un'interfaccia di dashboard che consenta all'Utente di visualizzare una panoramica dei record di transazione presenti nel registro, accedere ai dettagli dei singoli record, esportare e cancellare i record e, ove una transazione coinvolga una Relying Party, avviare una richiesta di cancellazione dei dati verso la relativa Relying Party (utilizzando, se disponibili, le informazioni di contatto registrate).

Il Fornitore del Wallet DEVE proteggere la riservatezza, l'integrità e l'autenticità del registro delle transazioni e di eventuali oggetti di log esportati in conformità a EUDI-TS 10; l'accesso al registro DEVE essere limitato all'Utente e all'Istanza del Wallet.

I record DEVONO essere conservati per almeno il periodo minimo previsto dalla normativa applicabile, come definito in *Politiche Generali di Conservazione dei Log* (pagina 671). Qualora limiti di spazio richiedano una cancellazione automatica, l'Istanza del Wallet DEVE notificare l'Utente tramite la dashboard, avvertire delle possibili conseguenze e offrire la possibilità di esportare i record interessati prima della cancellazione.

Esportazione e Cancellazione dei Record di Transazione

Esportazione: la dashboard DEVE consentire all'Utente di esportare uno o più record di transazione in un file utilizzando il formato comune e le misure di sicurezza definite in EUDI-TS 10.

Il file esportato DEVE poter essere salvato in una posizione di archiviazione esterna o remota scelta dall'Utente, tra le opzioni di archiviazione supportate dall'Istanza del Wallet.

Cancellazione: la dashboard DEVE consentire all'Utente di cancellare uno o più record di transazione dall'Istanza del Wallet in qualsiasi momento.

Prima di cancellare qualsiasi record di transazione, l'Istanza del Wallet DEVE mostrare all'Utente un avviso chiaro sulle possibili conseguenze per i diritti dell'Utente in materia di protezione dei dati.

La cancellazione dei record di transazione DEVE:

- applicarsi esclusivamente alle copie locali al Wallet dei record;
- essere irreversibile per l'Istanza del Wallet una volta completata.

L'Istanza del Wallet DEVE continuare a proteggere eventuali record di transazione rimanenti in conformità ai requisiti di riservatezza, integrità e autenticità definiti per il registro delle transazioni nella presente sezione.

Nessuna entità diversa dall'Utente PUÒ cancellare i record di transazione.

10.1.5 Backup e Ripristino

La funzionalità di **Backup e Ripristino** diventa rilevante quando l'Utente desidera cambiare dispositivo e/o Soluzione Wallet per accedere nuovamente alle Credenziali Elettroniche precedentemente emesse.

- Il dispositivo mobile è stato **perso**, **rubato**, **danneggiato** o **compromesso** (ad esempio, a causa di accesso non autorizzato).
- L'Utente sostituisce un'Istanza del Wallet esistente con una nuova istanza della stessa Soluzione Wallet.
- L'Utente ha cambiato il proprio dispositivo mobile e deve configurare la Soluzione Wallet sul nuovo dispositivo.
- L'Utente esegue un ripristino delle impostazioni di fabbrica sul telefono attuale e deve configurare nuovamente la Soluzione Wallet.

Nota

Per le Soluzioni Wallet basate sulle Specifiche Tecniche dell'IT Wallet, la migrazione verso una Soluzione Wallet diversa (nota come portabilità dei dati) può essere supportata seguendo la funzionalità di backup e ripristino descritta in questa sezione.

Flusso di Backup

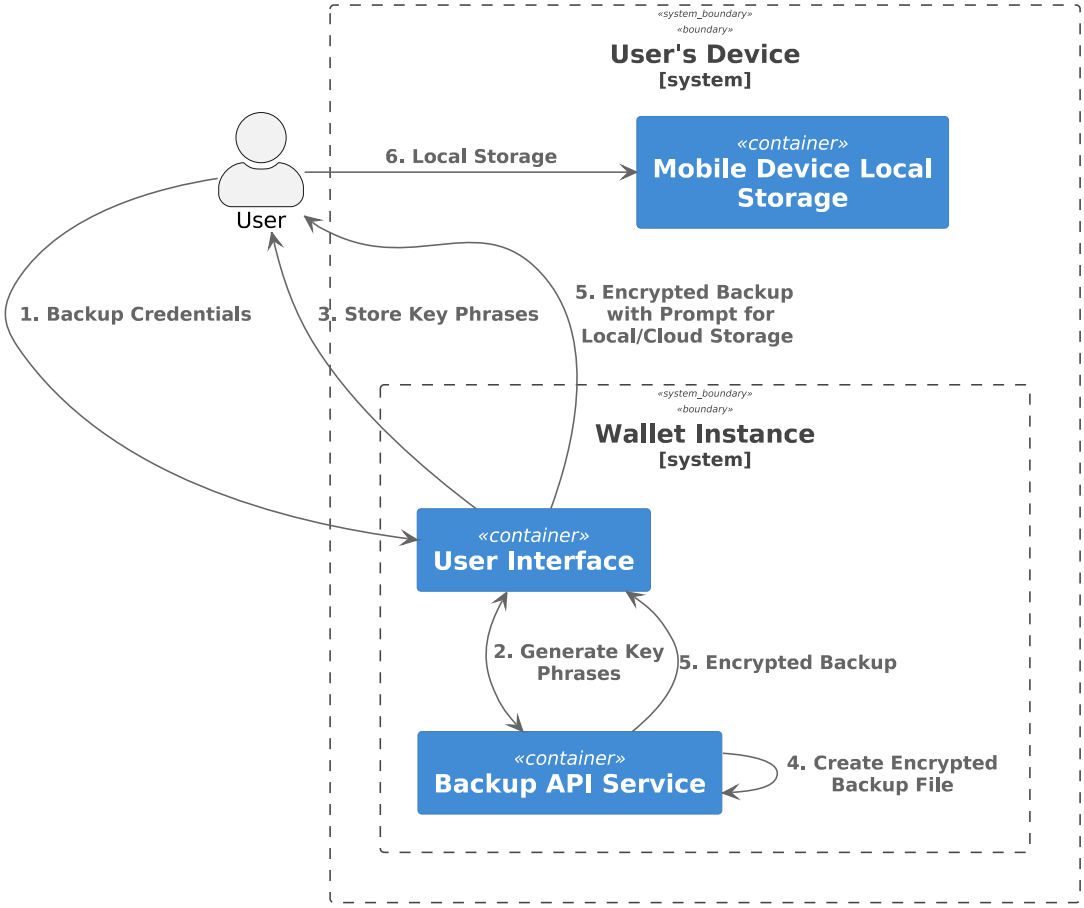


Fig. 10.8: Flusso di backup

Di seguito, la descrizione dei passaggi di Fig. 10.8:

Passaggio 1: L'Utente seleziona l'opzione per eseguire il backup delle Credenziali memorizzate nell'Istanza del Wallet (*WP_120* (pagina 812)).

Passaggi 2-3: L'Istanza del Wallet, utilizzando le API di backup, seleziona casualmente 10 frasi chiave da un elenco di parole pre-generato e le mostra all'Utente (*WP_120a* (pagina 812)). L'Utente DEVE conservare in modo sicuro la frase chiave scelta tra quelle proposte dal sistema (ad esempio, in un'app di gestione password) poiché sono fondamentali per il ripristino del backup (*WP_120b* (pagina 812)).

Nota

Come evidenziato nell'ARF, la crittografia è necessaria perché il file di backup è considerato sensibile. Anche se un attaccante conosce solo gli identificatori del Fornitore di Credenziali, può dedurre i diversi tipi di Credenziali Elettroniche, il che costituisce una violazione della privacy dell'Utente.

Nota

Per estrarre la chiave dall'elenco delle parole selezionate DEVE essere applicata una funzione di derivazione della chiave. Password-Based-Key-Derivation Function 2 (PBKDF2) è tra le più utilizzate basate su RFC 2898 ed è raccomandata dal NIST 800-132. Esistono anche altre tecniche rilevanti disponibili e ampiamente utilizzate, come Bcrypt, Scrypt e Argon2. Maggiori dettagli su questo approccio possono essere trovati qui (*WP_121* (pagina 812)).

Nota

Il livello di complessità per PBKDF2 è implementato attraverso un conteggio di iterazioni, che dovrebbe essere impostato diversamente in base all'algoritmo di hashing interno utilizzato. Il valore consigliato per l'algoritmo di hashing `SHA-256` è di 600000 iterazioni come indicato nell'OWASP Password Storage Cheatsheet (*WP_121a* (pagina 812)).

Passaggio 4: L'Istanza del Wallet esegue le seguenti operazioni per creare il file JWT di backup (*WP_122* (pagina 812)):

- Per ciascuna delle Credenziali con chiave vincolata all'hardware, aggiunge l'identificatore del Fornitore di Credenziali e il `credential_configuration_id` come voce nel JWT di backup (*WP_122d* (pagina 812)).
- Firma il JWT di backup utilizzando la chiave privata la cui chiave pubblica è attestata nella Wallet Instance Attestation. La relativa chiave pubblica attestata dal Fornitore di Wallet è fornita nella Wallet Instance Attestation (claim `cnf`). L'Istanza del Wallet DEVE verificare la validità della Wallet Instance Attestation prima di firmare il JWT di backup (*WP_123* (pagina 812)).
- Aggiunge il JWT di backup firmato come voce al file di backup (*WP_122* (pagina 812)).
- Cripta il file di backup utilizzando la frase chiave fornita (*WP_124* (pagina 812)).

Nota

Il JWT di Backup PUÒ contenere la cronologia delle transazioni per ogni voce di Credenziale all'interno del claim `credentials_backup` (*WP_122c* (pagina 812)).

Passaggio 5: All'Utente verrà richiesto di scegliere un'opzione di archiviazione per conservare in modo sicuro il file di backup. Le opzioni possono includere l'archiviazione nativa o soluzioni di archiviazione esterne, come l'archiviazione cloud, dispositivi USB, consegna via e-mail o altro (*WP_125* (pagina 812)).

Passaggio 6: Nel caso in cui l'Utente preferisca l'archiviazione nativa, il file di backup viene memorizzato sul dispositivo dell'Utente.

Un esempio non normativo dell'intestazione e del payload del JWT di backup è il seguente:

```
{
  "alg": "ES256",
  "typ": "wallet-unit-credentials-backup+jwt"
}
```

```
{
  "timestamp":"2024-12-13T16:35:06+01:00",
  "wallet_provider_id":"https://wallet-provider.example.org/",
  "wallet_instance_version":"v1.0",
  "wallet_instance_attestation":"eyJhbGciOiJFUzI1NiIsImVVfQz.eyJpc3MiOiAiaH...LCAibmJ",
  "credentials_backup": {
      "https://issuer.example.org/v1.0/mDL": ["org.iso.18013-5.1.mDL"],
      "https://eaa-provider.example.org/": ["dc_sd_jwt_EuropeanDisabilityCard"]
   }
}
```

L'intestazione JOSE del JWT di backup DEVE contenere i seguenti parametri OBBLIGATORI (*WP_122a* (pagina 812)):

Intestazione JOSE	Descrizione	Riferimento
alg	Un identificatore di algoritmo di firma digitale come da registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` o qualsiasi identificatore di algoritmo simmetrico (MAC).	**RFC 7516# section-4.1.1**.
typ	DEVE essere impostato su `wallet-unit-credentials-backup+jwt`	N/A

Il corpo del JWT di backup contiene i seguenti claim OBBLIGATORI (*WP_122b* (pagina 812)):

Claim	Descrizione
timestamp	Timestamp UNIX con l'ora di creazione del file di backup. Questo valore viene aggiornato ogni volta che una nuova voce di Credenziale viene aggiunta al file di backup.
wallet_provider_id	DEVE essere impostato sull'identificatore univoco del Fornitore di Wallet.
wallet_instance_version	DEVE essere impostato sulla versione della Soluzione Wallet di cui è stato eseguito il backup.
wallet_instance_attestation	DEVE essere impostato su un valore contenente il JWT della Wallet Instance Attestation.
credentials_backup	Oggetto che descrive le specifiche delle Credenziali di cui è stato eseguito il backup. Questo oggetto contiene un elenco di coppie nome/valore, dove ogni nome è un identificatore univoco del Fornitore di Credenziali. Questo identificatore viene utilizzato per avviare la fase di emissione. Il valore è un array di stringhe univoche. Ogni stringa corrisponde al `credential_configuration_id` che identifica la specifica Credenziale Elettronica che è stata emessa.

Flusso di ripristino per Credenziale con associazione hardware

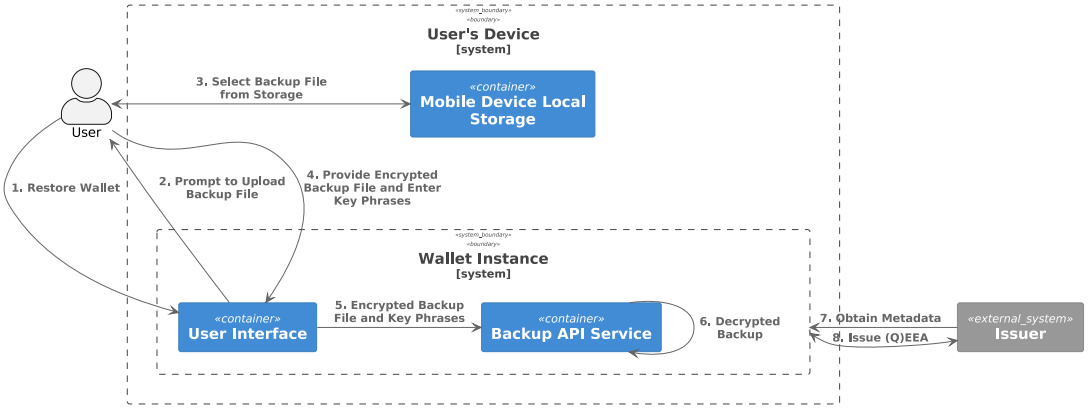


Fig. 10.9: Flusso di Ripristino

Considerando che l'Utente ha inizializzato la nuova Istanza del Wallet e questa è in stato attivo avendo ottenuto un nuovo Attestato Elettronico di Dati di Identificazione Personale, questa specifica allenta il requisito dell'ARF riguardante l'aggiunta dell'Attestato Elettronico di Dati di Identificazione Personale nel file di backup.

Allo stesso modo del PID, anche l'IT-Wallet ID NON DEVE essere incluso nel file di backup. Di seguito, la descrizione dei passaggi di Fig. 10.9:

Passaggi 1-6: L'Utente desidera ripristinare le Credenziali Elettroniche utilizzando il backup precedentemente creato con la propria Istanza del Wallet. L'Utente seleziona *ripristina backup delle Credenziali Elettroniche* nell'app dell'Istanza del Wallet e viene fornito all'Utente un prompt con la funzione di importazione. Il file di backup da importare può essere fornito utilizzando un

archivio locale o una posizione remota utilizzando anche un archivio cloud, e quindi inviare le frasi chiave di recupero (*WP_126* (pagina 812), *WP_127* (pagina 812)). Per verificare l'autenticità del file, l'Istanza del Wallet DEVE verificare la firma del JWT di backup per garantirne l'autenticità (*WP_129* (pagina 812)). Per fare ciò, estrae prima il JWT della Wallet Instance Attestation dal claim `wallet_instance_attestation` e ottiene la relativa chiave pubblica utilizzando la Wallet Instance Attestation (claim `cnf`) come specificato in *WP_128* (pagina 812).

Passaggi 7-8: L'Istanza del Wallet per ogni voce di Credenziale con associazione hardware nel payload del JWT di backup esegue i seguenti passaggi (*WP_130* (pagina 812)):

- Estrae l'identificatore del Fornitore di Credenziali e il `credential_configuration_id` dalla voce. Il primo viene utilizzato per identificare il Fornitore di Credenziali e ottenere i suoi metadati, mentre il secondo verrà utilizzato per segnalare il tipo di Credenziale al Fornitore di Credenziali (*WP_130a* (pagina 812)).
- Utilizzando l'identificatore del Fornitore di Credenziali, l'Istanza del Wallet ottiene i metadati del Fornitore di Credenziali e effettua una richiesta di emissione al Fornitore di Credenziali fornendo la nuova Associazione Crittografica con l'Utente (*WP_130b* (pagina 812)).

Nota

Durante il ripristino, l'Istanza del Wallet DEVE eseguire un Wallet-Initiated Authorization Code Issuance Flow come definito nella Sezione *Issuance Flow* (pagina 447), utilizzando una nuova Associazione Crittografica con l'Utente. Per le (Q)EAA, ciò include il gate di presentazione presso il Fornitore di Credenziali che agisce come Relying Party, secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente. Questo NON DEVE essere interpretato come il Re-issuance Flow definito nella Sezione *Re-issuance Flow* (pagina 473), che si applica solo all'aggiornamento della Credenziale sulla stessa istanza (`UPDATE` / `ATTRIBUTE_UPDATE`) e richiede un Refresh Token associato all'Istanza del Wallet originale. Dal punto di vista del Fornitore di Credenziali, il ripristino è indistinguibile da una prima emissione Wallet-Initiated. Le Credenziali ripristinate da Fornitori distinti RICHIEDONO un Authorization Code Issuance Flow separato per ciascun Fornitore.

Nota

L'Istanza del Wallet NON DEVE verificare la scadenza della Wallet Instance Attestation poiché il suo scopo principale è consentire all'Istanza del Wallet di verificare l'autenticità del file di backup assicurandosi che sia stato creato e firmato da un'Istanza del Wallet di uno specifico Fornitore di Wallet (*WP_128a* (pagina 812)).

10.1.6 Entity Configuration del Fornitore di Wallet

Una richiesta HTTP GET all'endpoint della Federazione consente di recuperare la Entity Configuration del Fornitore di Wallet (*WP_001* (pagina 778)).

La Entity Configuration del Fornitore di Wallet restituita DEVE contenere gli attributi descritti nelle sezioni seguenti.

La Entity Configuration del Fornitore di Wallet è un JWT firmato contenente le chiavi pubbliche e gli algoritmi supportati dalla Soluzione Wallet come componente del Fornitore di Wallet. È strutturata in conformità con OID-FED e con *L'Infrastruttura di Trust* (pagina 117) delineata in questa specifica (*WP_002* (pagina 778)).

Header JWT della Entity Configuration del Fornitore di Wallet

Key	Value
alg	Algoritmo utilizzato per verificare la firma del token. DEVE essere uno dei possibili valori indicati in *Algoritmi Crittografici* (pagina 647) (ad es., ES256).
kid	Impronta digitale della chiave pubblica utilizzata per la firma.
typ	Tipo di media, impostato su `entity-statement+jwt`.

Payload JWT della Entity Configuration del Fornitore di Wallet

Key	Value
`iss`	OBBLIGATORIO. URL pubblico della Solutione Wallet.
`sub`	OBBLIGATORIO. URL pubblico della Solutione Wallet.
`iat`	OBBLIGATORIO. Data e ora di emissione in formato Unix Timestamp.
`exp`	OBBLIGATORIO. Data e ora di scadenza in formato Unix Timestamp.
`authority_hints`	OBBLIGATORIO. Array di URL (String) contenente l'elenco degli URL delle Entità superiori immediate, come il Trust Anchor o un Intermediario, che POSSONO emettere un Entity Statement relativo alla Solutione Wallet.
`jwks`	OBBLIGATORIO. Un JSON Web Key Set (JWKS) che rappresenta la parte pubblica delle chiavi di firma dell'Entità di Federazione. La chiave privata corrispondente è utilizzata dalla Soluzione Wallet per firmare la Entity Configuration su se stessa.

continues on next page

Tabella 10.4 – continua dalla pagina precedente

Key	Value
`metadata`	OBBLIGATORIO. Oggetto JSON che rappresenta i Tipi di Entità e i metadati per quei Tipi di Entità. Ogni nome membro dell'oggetto JSON è un Identificatore di Tipo di Entità, e ogni valore DEVE essere un oggetto JSON contenente parametri di metadati secondo lo schema di metadati del Tipo di Entità. DEVE contenere i metadati `openid_wallet_provider` e OPZIONALMENTE i metadati `federation_entity`.

Nota

I test che coprono la struttura di Entity Configuration (header e payload) sono forniti in *WP_002a–002h* (pagina 778).

Esempio di Entity Configuration del Fornitore di Wallet

Di seguito è riportato un esempio non normativo di payload di una Entity Configuration del Fornitore di Wallet contenente metadati per

- *federation_entity*
- *openid_wallet_provider*


```
{
  "iss": "https://wallet-solution.example.org",
  "sub": "https://wallet-solution.example.org",
  "iat": 1687171759,
  "exp": 1709290159,
  "jwks": {
    "keys": [
      {
        "crv": "P-256",
        "kty": "EC",
        "x": "qrJrj3Af_B57sbOIRrcBM7br7wOc8ynj7lHFPTeffUk",
        "y": "1H0cWDyGgvU8w-kPKU_xycOCUNT2o0bwslIQtnPU6iM",
        "kid": "5t5YYpBhN-EgIEEI5iUzr6r0MR02LnVQ0OmekmNKcjY"
      }
    ]
  },
  "metadata": {
    "openid_wallet_provider": {
      "logo_uri": "https://wallet-solution.example.org/compact-logo.svg",
      "jwks": {
        "keys": [
          {
```

```
            "crv": "P-256",
            "kty": "EC",
            "x": "BxYsu3QvYmOz1fl1l5hGyPWlpvgTzz3AY3j3K_9zGPs",
            "y": "ob34Wmfah_ScQXaYMJWoBkZSwO-kQ0VTgMk4VZfu48w",
            "kid": "749b495837819c00cfee1749b495837819c00cfee1"
          }
        ]

      },
      "wallet_metadata": {
        "wallet_name": "Wallet X",
        "authorization_endpoint": "https://wallet-solution.digital-strategy.europa.eu/authorization",
        "credential_offer_endpoint": "https://wallet-solution.digital-strategy.europa.eu/credential_offer",
        "response_types_supported": [
          "vp_token"
        ],
        "vp_formats_supported": {
          "dc+sd-jwt": {
            "sd-jwt_alg_values": [
              "ES256",
              "ES384"
            ]
          }
        },
        "request_object_signing_alg_values_supported": [
          "ES256"
        ],
        "client_id_prefixes_supported": ["openid_federation", "x509_hash"]
      }
    },
    "federation_entity": {
      "organization_name": "IT-Wallet Provider",
      "homepage_uri": "https://wallet-provider.example.org",
      "policy_uri": "https://wallet-provider.example.org/privacy_policy",
      "tos_uri": "https://wallet-provider.example.org/info_policy",
      "logo_uri": "https://wallet-provider.example.org/logo.svg"
    }
  },
  "authority_hints": [
    "https://registry.eudi-wallet.example.it"
  ]
}
```

10.1.7 Metadati della Soluzione Wallet

L'oggetto JSON dei metadati la cui chiave è `openid_wallet_provider` contiene i seguenti parametri. Le chiavi pubbliche presenti in questo oggetto sono utilizzate esclusivamente per operazioni di firma e/o crittografia richieste a questa Entità quando agisce come componente del Fornitore di Wallet (ad esempio, firmare gli Attestati di Wallet per l'Istanza del Wallet).

Chiave	Valore
`jwks`	CONDIZIONALE. Documento JSON Web Key Set, passato per valore, contenente le chiavi dell'Entità per quel Tipo di Entità. DEVE essere presente se `jwks_uri` e `signed_jwks_uri` sono assenti.
`jwks_uri`	CONDIZIONALE. URL che fa riferimento a un documento JWK Set contenente le chiavi del Fornitore di Wallet per quel Tipo di Entità. Questo URL DEVE utilizzare lo schema https. DEVE essere presente se `jwks` e `signed_jwks_uri` sono assenti.
`signed_jwks_uri`	CONDIZIONALE. URL che fa riferimento a un JWT firmato avente come payload il documento JWK Set dell'Entità per quel Tipo di Entità. Questo URL DEVE utilizzare lo schema https. Il JWT DEVE essere firmato utilizzando una Chiave di Entità di Federazione. Una risposta positiva dall'URL DEVE utilizzare il codice di stato HTTP 200 con il Content Type `application/jwk-set+jwt`. DEVE essere presente se `jwks` e `jwks_uri` sono assenti.
`logo_uri`	OBBLIGATORIO. URL del logo dell'entità che verrà mostrato all'Utente durante le interazioni con l'istanza del Wallet. Il MIME type del logo DEVE essere `application/svg`.
`wallet_metadata`	OBBLIGATORIO. Contiene i parametri relativi ai metadati del Wallet come definiti nella *Tabella dei parametri dei metadati del portafoglio* (pagina 522) e seguenti due parametri • `credential_offer_endpoint` Credential Offer Endpoint del Wallet. • `wallet_name` Stringa contenente il nome del Wallet.

Nota

Alcuni flussi relativi agli Attestati Elettronici richiedono il recupero delle informazioni del Wallet prima di interagire con il Wallet stesso. Il flusso è descritto in *Flusso di Recupero dei Wallet Metadata* (pagina 570).

10.2 Soluzione del Fornitore di Attestati Elettronici

Un Fornitore di Attestati Elettronici, come Entità Organizzativa partecipante all'ecosistema IT-Wallet, DEVE fornire Soluzioni Tecniche che combinano software, hardware, servizi, impostazioni e configurazioni per emettere Attestati Elettronici alle Istanze del Wallet dell'Utente in modo sicuro e affidabile.

Il seguente diagramma illustra l'Architettura di Alto Livello della Soluzione del Fornitore di Attestati Elettronici.

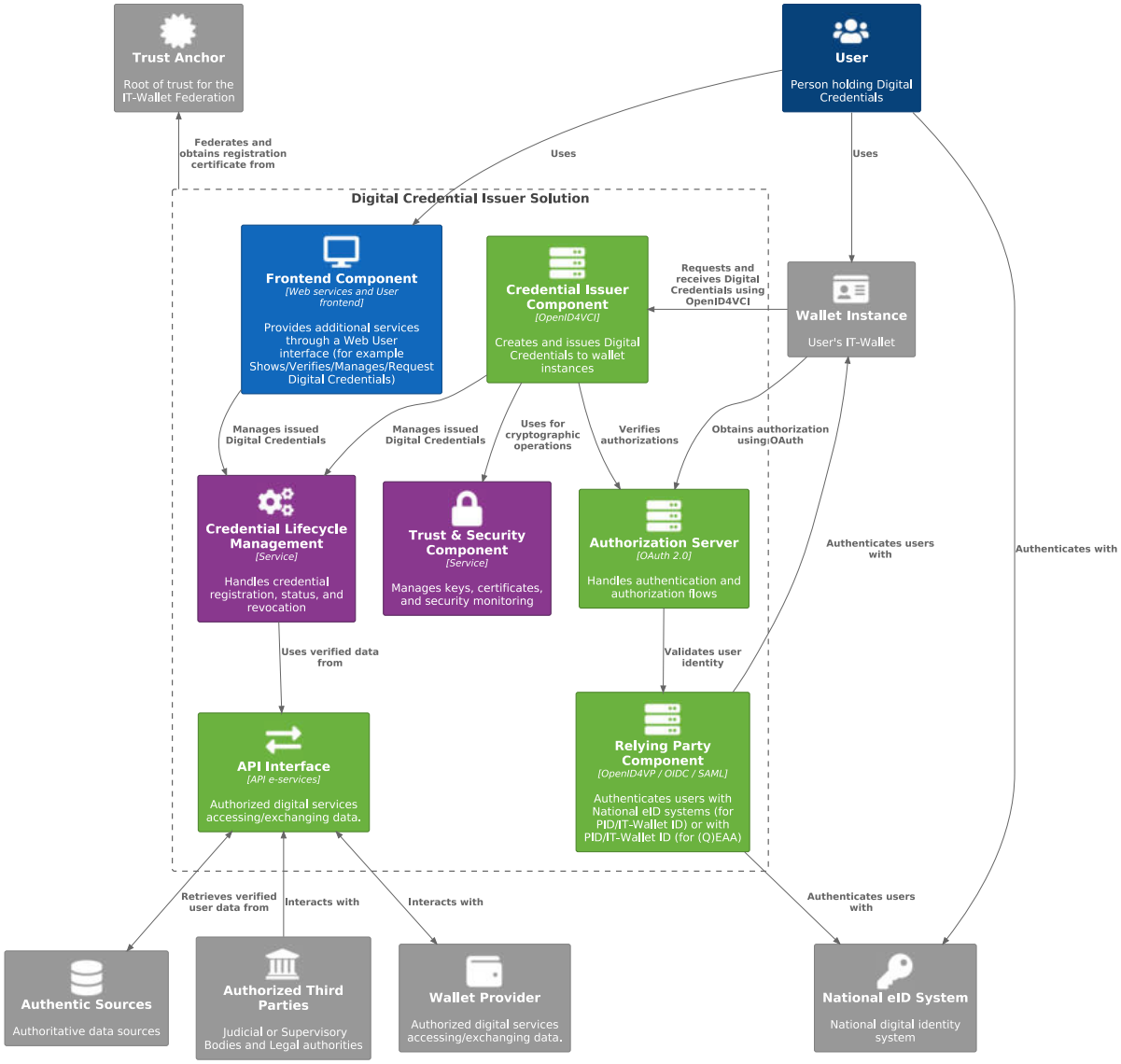


Fig. 10.10: Architettura di Alto Livello della Soluzione del Fornitore di Attestati Elettronici.

10.2.1 Requisiti del Fornitore di Attestati Elettronici

La Soluzione del Fornitore di Attestati Elettronici Elettronica DEVE:

1. Registrarsi presso l'Autorità di Federazione per ottenere la corretta autorizzazione per l'emissione di specifici tipi di Attestati Elettronici.
2. Implementare meccanismi sicuri di creazione ed emissione che garantiscano integrità e riservatezza.
3. Comunicare con le Fonti Autentiche attraverso Servizi API sicuri e affidabili per ottenere dati verificati dell'Utente.
4. Autenticarsi presso le Istanze del Wallet durante l'emissione per dimostrare la propria legittimità.
5. Supportare il flusso di emissione immediata (*immediate flow*) e PUÒ supportare l'emissione differita (*deferred flow*) per vari scenari operativi.
6. Implementare una gestione appropriata degli errori e notifiche all'Utente per tutti i processi.
7. Mantenere log di audit complete rispettando le normative sulla privacy.
8. Emettere Attestati Elettronici che supportano la Divulgazione Selettiva (*Selective Disclosure*).
9. Rinnovare periodicamente la conformità e l'aderenza alla Federazione IT-Wallet.
10. Registrare il Componente Relying Party all'interno dell'ecosistema di Federazione dell'Identità Digitale CIEid (per l'emissione di PID e IT-Wallet ID), e all'interno dell'ecosistema IT-Wallet (per l'emissione di (Q)EAA, se necessario).
11. Autenticare gli Utenti per l'emissione di PID, IT-Wallet ID e (Q)EAA secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.
12. Implementare procedure adeguate per l'intero ciclo di vita dell'Attestato Elettronico come dettagliato nella Sezione *Ciclo di Vita degli Attestati Elettronici* (pagina 422).

Per il Componente Frontend (se implementato):

13. Autenticare gli Utenti con un Livello di Garanzia (LoA) almeno pari a quello utilizzato per ottenere l'Attestato Elettronico che viene emesso o gestito.
14. Fornire misure di sicurezza appropriate per proteggere i dati dell'Utente e le informazioni dell'Attestato Elettronico.

10.2.2 Dettagli dei Componenti

Componente Frontend

Il Componente Frontend, se fornito dal Fornitore di Attestati Elettronici, DEVE fornire un'interfaccia Utente web per la gestione degli Attestati Elettronici, offrendo funzionalità per:

- Visualizzare e verificare gli Attestati Elettronici emessi e il loro stato di validità.
- Gestire il ciclo di vita degli Attestati Elettronici (ad es., revoca).
- Avviare l'emissione attraverso il meccanismo di *Credential Offer*.
- Fornire supporto e documentazione all'Utente.

I Fornitori di Attestati Elettronici POSSONO fornire servizi aggiuntivi all'Utente attraverso il Componente Frontend. Questi servizi aggiuntivi NON DEVONO essere in conflitto con i requisiti normativi o tecnici definiti in questa specifica tecnica o nelle normative nazionali/europee sulla sicurezza e la privacy.

Componente Credential Issuer

Sulla base della specifica OpenID4VCI e del profilo implementativo fornito nella Sezione *Emissione di Attestati Elettronici* (pagina 443), questo componente DEVE:

- Emettere Attestati Elettronici alle Istanze del Wallet.
- Elaborare le richieste di Attestati Elettronici.
- Ottenere i dati dell'Utente dalle Fonti Autentiche.
- Generare Attestati Elettronici correttamente formattati e firmati nei formati supportati (SD-JWT-VC, mDoc-CBOR). Vedere la Sezione *Modello di Dati degli Attestati Elettronici* (pagina 379) per maggiori dettagli.
- Implementare i protocolli e i flussi di emissione degli Attestati Elettronici.

Authorization Server

Questo componente basato su OAuth 2.0 DEVE:

- Gestire i flussi di autenticazione e autorizzazione.
- Gestire gli Access Token, Refresh e Authorization Code.
- Validare l'identità dell'Utente confermata dal Componente Relying Party.

Componente Relying Party

Questo componente DEVE autenticare gli Utenti, se richiesto secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.

Interfaccia API

Questo componente DEVE stabilire connessioni sicure con le Fonti Autentiche per:

- Recuperare dati verificati dell'Utente.
- Autenticare e autorizzare correttamente le richieste di connessione.
- Formattare i dati secondo gli schemi degli Attestati Elettronici.
- Fornire prove crittografiche dell'autenticazione dell'Utente quando richiesto.

Nota

Per le Fonti Autentiche pubbliche, un Fornitore di Attestati Elettronici DEVE utilizzare PDND secondo quanto definito nelle Sezioni *e-Service PDND* (pagina 714), *Aggiornamento dello Stato da parte delle Fonti Autentiche* (pagina 432), e *Catalogo degli e-Service PDND delle Fonti Autentiche* (pagina 634).

Gestione del Ciclo di Vita degli Attestati Elettronici

Questo componente DEVE gestire:

- Lo stato di validità degli Attestati Elettronici (ad esempio aggiornamenti, revoche o sospensioni).
- I processi e i flussi di revoca (implementazione di meccanismi per revocare o sospendere gli Attestati Elettronici), secondo quando definito nella Sezione *Ciclo di Vita degli Attestati Elettronici* (pagina 422).
- I processi e i flussi di rinnovo degli Attestati Elettronici, secondo i meccanismi definiti nella Sezione *Emissione di Attestati Elettronici* (pagina 443).

Componente Trust & Security

Questo componente DEVE garantire la sicurezza attraverso:

- Gestione delle chiavi e dei certificati.
- Log di audit.
- Monitoraggio della sicurezza e risposta agli incidenti.
- Conformità ai requisiti di sicurezza della Federazione IT-Wallet.

10.2.3 Modelli di Interazione

La Soluzione del Fornitore degli Attestati Elettronici supporta questi modelli di interazione:

1. **Utente verso Frontend**: Interazioni basate sul web per la gestione degli Attestati Elettronici.
2. **Frontend verso Fornitore degli Attestati Elettronici**: Converte le richieste dell'Utente in messaggi del protocollo OpenID4VCI.
3. **Istanza del Wallet verso Fornitore degli Attestati Elettronici**: Interazioni dirette basate sul protocollo seguendo il flusso di emissione.
4. **Relying Party verso Gestori di Identità**: Interazioni di autenticazione con i sistemi nazionali di Identità Digitale o verifica del PID/IT-Wallet ID.
5. **Interfaccia API verso Fonti Autentiche**: Chiamate API sicure per recuperare dati verificati dell'Utente.

Tutte le interazioni devono seguire le considerazioni di sicurezza nella Sezione *Emissione di Attestati Elettronici* (pagina 443), inclusa la corretta gestione di token e materiali crittografici.

10.2.4 Entity Configuration del Fornitore di Attestati Elettronici

I Fornitori di Attestati Elettronici, in quanto Entità di Federazione, DEVONO rispettare le linee guida delineate nella Sezione *Configurazione della Federazione* (pagina 125). In particolare, DEVONO fornire un endpoint *well-known* che ospiti la loro Entity Configuration. Quest'ultima DEVE contenere i parametri definiti nelle Sezioni *Entity Configuration Foglie e Intermediari* (pagina 130) e *Parametri Comuni delle Entity Configuration* (pagina 127).

I Fornitori di Attestati Elettronici DEVONO fornire i seguenti Metadata:

- *federation_entity*
- *oauth_authorization_server*
- *openid_credential_issuer*

Nei casi in cui i Fornitori di Attestati Elettronici di Attributi autenticano gli Utenti utilizzando la loro Istanza del Wallet, allora i Metadata per *openid_credential_verifier* DEVONO essere forniti in aggiunta ai Metadata sopra indicati. Nel caso in cui uno schema nazionale di Identità Digitale sia utilizzato dal Fornitore di Attestati Elettronici di Dati di Identificazione Personale per l'autenticazione dell'Utente, essi POSSONO includere i Metadata per *openid_relying_party* all'interno della loro Entity Configuration. I Metadata *openid_relying_party* DEVONO essere conformi alle Regole Tecniche SPID/CIE-OpenID-Connect-Specifications.

I Metadata *federation_entity* DEVONO contenere i parametri come definiti nella Sezione *Metadati di federation_entity Foglie* (pagina 131).

I Metadata *oauth_authorization_server* DEVONO contenere i parametri come definiti nella Sezione credential-issuer-metadata:Metadata per oauth_authorization_server.

I Metadata *openid_credential_issuer* DEVONO contenere i parametri come definiti nella Sezione credential-issuer-metadata:Metadata per openid_credential_issuer.

I Metadata *openid_credential_verifier* DEVONO contenere i parametri come definiti nella Sezione *Entity Configuration Relying Party* (pagina 372).

Esempio di Entity Configuration di un Fornitore di Attestati Elettronici di Attributi

Di seguito è riportato un esempio non normativo di una Entity Configuration di un Fornitore di Attestati Elettronici di Attributi contenente Metadata per:

- *federation_entity*
- *oauth_authorization_server*
- *openid_credential_issuer*
- *openid_credential_verifier*


```
{
    "iat": 1718207217,
    "exp": 1749743216,
    "iss": "https://eaa-provider.example.org",
    "sub": "https://eaa-provider.example.org",
    "authority_hints": [
        "https://trust-anchor.example.org"
    ],
    "jwks": {
        "keys": [
            {
                "kid": "FANFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
```

```
            "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
        }
    ]
},
"metadata": {
    "federation_entity": {
        "homepage_uri": "https://eaa-provider.example.org/",
        "organization_name": "Organization Name",
        "contacts": [
            "informazioni@example.it",
            "protocollo@pec.example.it"
        ],
        "tos_uri": "https://eaa-provider.example.org/public/info_policy.html",
        "policy_uri": "https://eaa-provider.example.org/public/privacy_policy.html",
        "logo_uri": "https://eaa-provider.example.org/public/logo.svg"
    },
    "oauth_authorization_server": {
        "issuer": "https://eaa-provider.example.org",
        "pushed_authorization_request_endpoint": "https://eaa-provider.example.org/as/par",
        "authorization_endpoint": "https://eaa-provider.example.org/authorize",
        "token_endpoint": "https://eaa-provider.example.org/token",
        "client_registration_types_supported": [
            "automatic"
        ],
        "code_challenge_methods_supported": [
            "S256"
        ],
        "acr_values_supported": [
            "https://trust-anchor.example.it/loa/substantial",
            "https://trust-anchor.example.it/loa/high"
        ],
        "scopes_supported": [
            "EuropeanDisabilityCard",
            "mDL"
        ],
        "response_types_supported": [
            "code"
        ],
        "authorization_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "grant_types_supported": [
            "authorization_code"
        ],
        "token_endpoint_auth_methods_supported": [
```

```
            "attest_jwt_client_auth"
        ],
        "client_attestation_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "client_attestation_pop_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "token_endpoint_auth_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "request_object_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "dpop_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "jwks": {
            "keys": [
                {
                    "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                    "kty": "EC",
                    "crv": "P-256",
                    "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                    "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                }
            ]
        }
    },
    "openid_credential_issuer": {
        "credential_issuer": "https://eaa-provider.example.org",
        "credential_endpoint": "https://eaa-provider.example.org/credential",
        "nonce_endpoint": "https://eaa-provider.example.org/nonce-endpoint",
        "deferred_credential_endpoint": "https://eaa-provider.example.org/deferred-credenti
al",
        "notification_endpoint": "https://eaa-provider.example.org/notification",
        "display": [
```

```
                {
                    "name": "EAA Provider",
                    "locale": "it-IT",
                    "logo": {
                        "uri": "https://eaa-provider.example.org/public/compact-logo-IT.svg",
                        "uri#integrity": "2Rltygo7bKBE4E498tb/lLn9DIkimBhrzrsUERyUQjo=",
                        "alt_text": "Testo alternativo all'immagine del logo"
                    }

                },
                {
                    "name": "EAA Provider",
                    "locale": "en-US",
                    "logo": {
                        "uri": "https://eaa-provider.example.org/public/compact-logo-US.svg",
                        "uri#integrity": "mBQEi42aj0VYnFngpOjbqJGkuCnB5LmbZRnWpO+AHqUBVQxOe9kMI
POz+pSjRArE",
                        "alt_text": "Alternative text for the logo image"
                    }
                }
            ],
            "credential_configurations_supported": {
                "dc_sd_jwt_EuropeanDisabilityCard": {
                    "format": "dc+sd-jwt",
                    "scope": "EuropeanDisabilityCard",
                    "cryptographic_binding_methods_supported": [
                        "jwk"
                    ],
                    "credential_signing_alg_values_supported": [
                        "ES256",
                        "ES384",
                        "ES512"
                    ],
                    "proof_types_supported": {
                        "jwt": {
                            "proof_signing_alg_values_supported": [
                                "ES256",
                                "ES384",
                                "ES512"
                            ],
                            "key_attestations_required": {
                            "key_storage": [ "iso_18045_enhanced-basic" ],
                            "user_authentication": [ "iso_18045_enhanced-basic" ]
                            }
                        }
                    },
                    "vct": "urn:eudi:EuropeanDisabilityCard:it:1",
```

```
                    "schema_id": "EuropeanDisabilityCard+dc+sd-jwt+urn:eudi:EuropeanDisabilityCard:it:1",
                    "authentic_sources": [
                        {
                            "entity_id": "https://authentic-source.example.com",
                            "dataset_id": "12345"
                        }

                    ],
                    "credential_metadata": {
                        "display": [
                            {
                                "name": "Carta della disabilità europea",
                                "locale": "it-IT",
                                "description": "Versione digitale della carta della disabilità europea",
                                "logo": {
                                    "uri": "https://trust-registry.it-wallet.example.it/logos/EuropeanDisabilityCard/logo-IT.svg",
                                    "uri#integrity": "3GGZJH7igkyeprXI9Plm3dQsWMYOeuz0MaEK34PC7sOtLfgeoK9scXOvMeiACF8Tk2WOT36NsYCPduoLyOD1Sg==",
                                    "alt_text": "Testo alternativo all'immagine del logo"
                                },
                                "background_color": "#12107c"
                            },
                            {
                                "name": "European Disability Card",
                                "locale": "en-US",
                                "description": "Digital version of the European Disability Card",
                                "logo": {
                                    "uri": "https://trust-registry.it-wallet.example.it/logos/EuropeanDisabilityCard/logo-US.svg",
                                    "uri#integrity": "3GGZJH7igkyeprXI9Plm3dQsWMYOeuz0MaEK34PC7sOtLfgeoK9scXOvMeiACF8Tk2WOT36NsYCPduoLyOD1Sg==",
                                    "alt_text": "Alternative text for the logo image"
                                },
                                "background_color": "#12107c"
                            }
                        ],
                        "claims": [
                            {
                                "path": ["document_number"],
                                "mandatory": "true",
                                "sd": "always",
                                "display": [
                                    {
```

```
                                    "name": "Numero Documento",
                                    "locale": "it-IT",
                                    "description": "Numero Documento della carta europea de
lla disabilità"
                                },
                                {
                                    "name": "Document Number",
                                    "locale": "en-US",
                                    "description": "Document number of the European Disabil
ity Card"
                                }
                            ]
                        },
                        {
                            "path": ["given_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Nome",
                                    "locale": "it-IT",
                                    "description": "Nome del titolare della carta"
                                },
                                {
                                    "name": "Name",
                                    "locale": "en-US",
                                    "description": "Cardholder name"
                                }
                            ]
                        },
                        {
                            "path": ["family_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Cognome",
                                    "locale": "it-IT",
                                    "description": "Cognome del titolare della carta"
                                },
                                {
                                    "name": "Family Name",
                                    "locale": "en-US",
                                    "description": "cardholder last name"
                                }
                            ]
                        },
```

```
                {
                    "path": ["birth_date"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Data di Nascita (YYYY-MM-GG)",
                            "locale": "it-IT",
                            "description": "Data di nascita del titolare della cart
a"

                        },
                        {
                            "name": "Date of Birth (YYYY-MM-GG)",
                            "locale": "en-US",
                            "description": "cardholder birth date"
                        }
                    ]
                },
                {
                    "path": ["personal_administrative_number"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Codice Fiscale",
                            "locale": "it-IT",
                            "description": "Codice fiscale del titolare della carta
"

                        },
                        {
                            "name": "Tax Identification Number",
                            "locale": "en-US",
                            "description": "cardholder tax identification number"
                        }
                    ]
                },
                {
                    "path": ["expiry_date"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Data di Scadenza (YYYY-MM-GG)",
                            "locale": "it-IT",
                            "description": "Data di scadenza della carta"
                        },
```

```
                                {
                                    "name": "Expiration Date (YYYY-MM-GG)",
                                    "locale": "en-US",
                                    "description": "Card expiration date"
                                }
                            ]
                        },
                        {
                            "path": ["constant_attendance_allowance"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Diritto accompagnatore",
                                    "locale": "it-IT",
                                    "description": "Questo campo indica se il titolare dell
a carta ha diritto all'indennità di accompagnamento"
                                },
                                {
                                    "name": "Constant attendance allowance",
                                    "locale": "en-US",
                                    "description": "This field indicates whether the cardho
lder is entitled to the accompanying allowance"
                                }
                            ]
                        },
                        {
                            "path": ["portrait"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Foto",
                                    "locale": "it-IT",
                                    "description": "Fotografia del titolare della carta"

                                },
                                {
                                    "name": "Portrait",
                                    "locale": "en-US",
                                    "description": "Photo of the cardholder"
                                }
                            ]
                        },
                        {
                            "path": ["link_qr_code"],
                            "mandatory": "true",
```

```
                    "sd": "always",
                    "display": [
                        {
                            "name": "Link QR Code",
                            "locale": "it-IT",
                            "description": "Questo campo contiene il QR-Code"
                        },
                        {
                            "name": "Link QR Code",
                            "locale": "en-US",
                            "description": "This field contains the QR-Code"
                        }
                    ]
                }
            ]
        }
    },
    "dc_sd_jwt_mDL": {
        "format": "dc+sd-jwt",
        "scope": "mDL",
        "cryptographic_binding_methods_supported": [
            "jwk"
        ],
        "credential_signing_alg_values_supported": [
            "ES256",
            "ES384",
            "ES512"
        ],
        "proof_types_supported": {
            "jwt": {
                "proof_signing_alg_values_supported": [
                    "ES256",
                    "ES384",
                    "ES512"
                ]
            }
        },
        "vct": "urn:eudi:mDL:it:1",
        "schema_id": "mDL+dc+sd-jwt+urn:eudi:mDL:it:1",
        "authentic_sources": [
            {
                "entity_id": "https://authentic-source.example.com",
                "dataset_id": "12345"
            }

        ],
        "credential_metadata": {
```

```
                    "display": [
                        {
                            "name": "Patente di guida",
                            "locale": "it-IT",
                            "description": "Versione digitale della patente di guida",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-IT.svg",
                                "uri#integrity": "qU4OnSrnRcL695rQqy9ayCMx6sboaXxEaW5rXvSAm
nQ=",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        },
                        {
                            "name": "Mobile Driver's License",
                            "locale": "en-US",
                            "description": "Digital version of Mobile Driver's License",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-US.svg",
                                "uri#integrity": "F5P1wXpfToammXkx6HF++66G534ystjOy7r4UMWwz
1lcZYJU8kAKRKtFua7DvZLe",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        }
                    ],
                    "claims": [
                        {
                            "path": ["given_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Nome",
                                    "locale": "it-IT",
                                    "description": "Nome del titolare della patente"
                                },
                                {
                                    "name": "First Name",
                                    "locale": "en-US",
                                    "description": "Name of the driving license holder"
                                }
                            ]
                        },
                        {
```

```
                "path": ["family_name"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Cognome",
                        "locale": "it-IT",
                        "description": "Cognome del titolare della patente"
                    },
                    {
                        "name": "Family Name",
                        "locale": "en-US",
                        "description": "last name of the driving license holder
"
                    }
                ]
            },
            {
                "path": ["birth_date"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Data di nascita (YYYY-MM-GG)",
                        "locale": "it-IT",
                        "description": "Data di nascita del titolare della pate
nte"
                    },
                    {
                        "name": "Date of Birth (YYYY-MM-GG)",
                        "locale": "en-US",
                        "description": "Date of birth of the driving license ho
lder"
                    }
                ]
            },
            {
                "path": ["place_of_birth"],
                "mandatory": "true",
                "sd": "always",
                "display": [
                    {
                        "name": "Luogo di Nascita",
                        "locale": "it-IT",
                        "description": "Luogo di nascita del titolare della pat
ente"
                    },
```

```
                                {
                                    "name": "Place of Birth",
                                    "locale": "en-US",
                                    "description": "Place of birth of the driving license h
older"
                                }
                            ]
                        },
                        {
                            "path": ["issue_date"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Data di rilascio (YYYY-MM-GG)",
                                    "locale": "it-IT",
                                    "description": "Data di rilascio della patente"
                                },
                                {
                                    "name": "Issue Date (YYYY-MM-GG)",
                                    "locale": "en-US",
                                    "description": "Date of issuance of the driving license
"
                                }
                            ]
                        },
                        {
                            "path": ["expiry_date"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Data di scadenza (YYYY-MM-GG)",
                                    "locale": "it-IT",
                                    "description": "Data di scadenza della patente"
                                },
                                {
                                    "name": "Expiry Date (YYYY-MM-GG)",
                                    "locale": "en-US",
                                    "description": "Date of expiry of the driving license"
                                }
                            ]
                        },
                        {
                            "path": ["issuing_country"],
                            "mandatory": "true",
                            "sd": "always",
```

```
            "display": [
                {
                    "name": "Paese di rilascio",
                    "locale": "it-IT",
                    "description": "Paese di rilascio della patente"

                },
                {
                    "name": "Issuing Country",
                    "locale": "en-US",
                    "description": "Issuing country of the driving license"
                }
            ]
        },
        {
            "path": ["issuing_authority"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Autorità di rilascio",
                    "locale": "it-IT",
                    "description": "Autorità che ha rilasciato la patente"
                },
                {
                    "name": "Issuing Authority",
                    "locale": "en-US",
                    "description": "Issuing authority of the driving licens
e"
                }
            ]
        },
        {
            "path": ["document_number"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Numero di documento",
                    "locale": "it-IT",
                    "description": "Numero Documento della patente"
                },
                {
                    "name": "Document Number",
                    "locale": "en-US",
                    "description": "Document number of the driving license"
                }
```

```
            ]
        },
        {
            "path": ["portrait"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Foto codificata",
                    "locale": "it-IT",
                    "description": "Fotografia del titolare della patente"
                },
                {
                    "name": "Portrait base64 encoded",
                    "locale": "en-US",
                    "description": "Photo of of the driving licence holder"
                }
            ]
        },
        {
            "path": ["driving_privileges"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Abilitazioni alla guida",
                    "locale": "it-IT",
                    "description": "Elenco delle categorie di abilitazione"
                },
                {
                    "name": "Driving Privileges",
                    "locale": "en-US",
                    "description": "Driving Privileges list"
                }
            ]
        },
        {
            "path": ["restrictions_conditions"],
            "mandatory": "true",
            "sd": "always",
            "display": [
                {
                    "name": "Annotazioni/Restrizioni",
                    "locale": "it-IT",
                    "description": "Annotazioni/Restrizioni valide per tutt
e le categorie separate da spazio"
                },
```

```
                                {
                                    "name": "Restrictions/Conditions",
                                    "locale": "en-US",
                                    "description": "Restriction/Condition for all driving p
rivileges separated by space"
                                }
                            ]
                        },
                        {
                            "path": ["driving_privileges_details"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Dettagli delle categorie di abilitazione",
                                    "locale": "it-IT",
                                    "description": "Dettagli relativi alle specifiche categ
orie di abilitazione"
                                },
                                {
                                    "name": "Driving privilege details",
                                    "locale": "en-US",
                                    "description": "Details related to the specific driving
privileges"
                                }
                            ]
                        }
                    ]
                }
            },
            "mso_mdoc_mDL": {
                "format": "mso_mdoc",
                "scope": "mDL",
                "doctype": "org.iso.18013.5.1.mDL",
                "schema_id": "mDL+mso_mdoc+org.iso.18013.5.1.mDL",
                "authentic_sources": [
                    {
                        "entity_id": "https://authentic-source.example.com",
                        "dataset_id": "12345"
                    }

                ],
                "cryptographic_binding_methods_supported": [
                    "cose_key"
                ],
                "credential_signing_alg_values_supported": [
                    -7,
```

```
                    -35,
                    -36
                ],
                "credential_metadata": {
                    "display": [
                        {
                            "name": "Patente di guida",
                            "locale": "it-IT",
                            "description": "Versione digitale della patente di guida",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-IT.svg",
                                "uri#integrity": "qU4OnSrnRcL695rQqy9ayCMx6sboaXxEaW5rXvSAm
nQ=",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        },
                        {
                            "name": "Mobile Driver's License",
                            "locale": "en-US",
                            "description": "Digital version of Mobile Driver's License",
                            "logo": {
                                "uri": "https://trust-registry.it-wallet.example.it/logos/m
DL/logo-US.svg",
                                "uri#integrity": "F5P1wXpfToammXkx6HF++66G534ystjOy7r4UMWwz
1lcZYJU8kAKRKtFua7DvZLe",
                                "alt_text": "Testo alternativo all'immagine del logo"
                            },
                            "background_color": "#12107c"
                        }
                    ],
                    "claims": [
                        {
                            "path": ["org.iso.18013.5.1", "given_name"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Nome",
                                    "locale": "it-IT",
                                    "description": "Nome del titolare della patente"
                                },
                                {
                                    "name": "First Name",
                                    "locale": "en-US",
                                    "description": "Name of the driving license holder"
```

```
                        }
                    ]
                },
                {
                    "path": ["org.iso.18013.5.1", "family_name"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Cognome",
                            "locale": "it-IT",
                            "description": "Cognome del titolare della patente"
                        },
                        {
                            "name": "Family Name",
                            "locale": "en-US",
                            "description": "last name of the driving license holder
"
                        }
                    ]
                },
                {
                    "path": ["org.iso.18013.5.1", "birth_date"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Data di nascita (YYYY-MM-GG)",
                            "locale": "it-IT",
                            "description": "Data di nascita del titolare della pate
nte"
                        },
                        {
                            "name": "Date of Birth (YYYY-MM-GG)",
                            "locale": "en-US",
                            "description": "Date of birth of the driving license ho
lder"
                        }
                    ]
                },
                {
                    "path": ["org.iso.18013.5.1", "birth_place"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Luogo di Nascita",
```

```
                                    "locale": "it-IT",
                                    "description": "Luogo di nascita del titolare della pat
ente"
                                },
                                {
                                    "name": "Place of Birth",
                                    "locale": "en-US",
                                    "description": "Place of birth of the driving license h
older"
                                }
                            ]
                        },
                        {
                            "path": ["org.iso.18013.5.1", "issue_date"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Data di rilascio (YYYY-MM-GG)",
                                    "locale": "it-IT",
                                    "description": "Data di emissione della patente"
                                },
                                {
                                    "name": "Issue Date (YYYY-MM-GG)",
                                    "locale": "en-US",
                                    "description": "Date of issuance of the driving license
"
                                }
                            ]
                        },
                        {
                            "path": ["org.iso.18013.5.1", "expiry_date"],
                            "mandatory": "true",
                            "sd": "always",
                            "display": [
                                {
                                    "name": "Data di scadenza (YYYY-MM-GG)",
                                    "locale": "it-IT",
                                    "description": "Data di scadenza della patente"
                                },
                                {
                                    "name": "Expiry Date (YYYY-MM-GG)",
                                    "locale": "en-US",
                                    "description": "Date of expiry of the driving license"
                                }
                            ]
                        },
```

```
                {
                    "path": ["org.iso.18013.5.1", "issuing_country"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Paese di rilascio",
                            "locale": "it-IT",
                            "description": "Paese di rilascio della patente"
                        },
                        {
                            "name": "Issuing Country",
                            "locale": "en-US",
                            "description": "Issuing country of the driving license"
                        }
                    ]
                },
                {
                    "path": ["org.iso.18013.5.1", "issuing_authority"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Autorità di rilascio",
                            "locale": "it-IT",
                            "description": "Autorità che ha rilasciato la patente"
                        },
                        {
                            "name": "Issuing Authority",
                            "locale": "en-US",
                            "description": "Issuing authority of the driving licens
e"
                        }
                    ]
                },
                {
                    "path": ["org.iso.18013.5.1", "document_number"],
                    "mandatory": "true",
                    "sd": "always",
                    "display": [
                        {
                            "name": "Numero di documento",
                            "locale": "it-IT",
                            "description": "Numero Documento della patente"
                        },
                        {
                            "name": "Document Number",
```

```
                "locale": "en-US",
                "description": "Document number of the driving license"
            }
        ]
    },
    {
        "path": ["org.iso.18013.5.1", "portrait"],
        "mandatory": "true",
        "sd": "always",
        "display": [
            {
                "name": "Foto codificata",
                "locale": "it-IT",
                "description": "Fotografia del titolare della patente"
            },
            {
                "name": "Portrait",
                "locale": "en-US",
                "description": "Photo of of the driving licence holder"
            }
        ]
    },
    {
        "path": ["org.iso.18013.5.1", "driving_privileges"],
        "mandatory": "true",
        "sd": "always",
        "display": [
            {
                "name": "Abilitazioni alla guida",
                "locale": "it-IT",
                "description": "Elenco delle categorie di abilitazione"
            },
            {
                "name": "Driving Privileges",
                "locale": "en-US",
                "description": "Driving Privileges list"
            }
        ]
    },
    {
        "path": ["org.iso.18013.5.1", "un_distinguishing_sign"],
        "mandatory": "true",
        "sd": "always",
        "display": [
            {
                "name": "Codice identificativo della Nazione",
                "locale": "it-IT",
```

```
                                    "description": "Segno distintivo del paese che emette l
a patente"
                                },
                                {
                                    "name": "Distinguishing sign of the issuing country",
                                    "locale": "en-US",
                                    "description": "Distinguishing sign of the issuing coun
try"
                                }
                            ]
                        }
                    ]
                }
            }
        },
        "jwks": {
            "keys": [
                {
                    "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                    "kty": "EC",
                    "crv": "P-256",
                    "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                    "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
                }
            ]
        },
        "trust_frameworks_supported": [
            "it_cie",
            "it_wallet",
            "eudi_wallet"
        ],
        "status_list_aggregation_endpoint": "https://eaa-provider.example.org/statuslists/a
ggregation"
    },
    "openid_credential_verifier": {
        // see relying party metadata section and endpoints
    },
    "jwks": {
        "keys": [
            {
                "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
            }
        ]
```

```
        }
    }
}
}
```

10.2.5 Metadata del Fornitore di Attestati Elettronici

Metadata per oauth_authorization_server

I Metadata *oauth_authorization_server* DEVONO contenere i seguenti parametri.

Claim	Descrizione
issuer	DEVE contenere un URL HTTPS che identifica in modo univoco il Fornitore di Attestati Elettronici.
pushed_authorization_request_endpoint	L'URL dell'endpoint della *Pushed Authorization Request* dove un'Istanza del Wallet DEVE inviare una *authorization request* per ottenere un valore *request_uri*, che può quindi essere utilizzato all'*authorization endpoint*. Vedi **RFC 9126#as_metadata**.
authorization_endpoint	URL dell'*authorization endpoint* dell'*authorization server*. Vedi **RFC 8414#section-2**.
token_endpoint	URL del *token endpoint* dell'*authorization server*. Vedi **RFC 8414#section-2**.
client_registration_types_supported	Array che specifica i *registration types* supportati. L'*authorization server* DEVE supportare *automatic*. Vedi OID-FED Sezione 5.1.3.
code_challenge_methods_supported	Array JSON contenente un elenco di metodi di *code challenge* previsti da Proof Key for Code Exchange (PKCE) **RFC 7636** e supportati dall'*authorization server*. L'*authorization server* DEVE supportare *S256*.
acr_values_supported	Vedi OpenID Connect Discovery 1.0 Section 3. I valori supportati sono: • *https://{Trust Anchor Domain}/loa/low* • *https://{Trust Anchor Domain}/loa/substantial* • *https://{Trust Anchor Domain}/loa/high*
scopes_supported	Array JSON contenente un elenco dei valori *scope* supportati. Vedi **RFC 8414#section-2**.
response_types_supported	Array JSON contenente un elenco dei valori "*response_type*" supportati, come specificato in **RFC 8414**. Il valore supportato DEVE essere *code*.
authorization_signing_alg_values_supported	Array JSON contenente un elenco degli algoritmi di firma supportati **RFC 7515** (valori *alg*). I valori DEVONO essere impostati secondo la Sezione *Algoritmi Crittografici* (pagina 647).
grant_types_supported	Array JSON contenente un elenco dei valori di *grant type* supportati. L'*authorization server* DEVE supportare *authorization_code*.

continues on next page

Tabella 10.6 – continua dalla pagina precedente

Claim	Descrizione
token_endpoint_ auth_methods_ supported	Array JSON contenente un elenco dei metodi di *client authentication* supportati. Il *token endpoint* DEVE supportare *attest_jwt_client_auth* come definito in OAUTH-ATTESTATION-CLIENT-AUTH.
client_attestation_ signing_alg_values_ supported	Array JSON con l'elenco dei valori JWS "alg" supportati per la Wallet Attestation (`oauth-client-attestation+jwt`). I valori DEVONO provenire dalla Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVONO includere `none` né algoritmi simmetrici (MAC).
client_attestation_ pop_signing_alg_ values_supported	Array JSON con l'elenco dei valori JWS "alg" supportati per la Proof-of-Possession della Wallet Attestation (`oauth-client-attestation-pop+jwt`). I valori DEVONO provenire dalla Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVONO includere `none` né algoritmi simmetrici (MAC).
token_endpoint_ auth_signing_alg_ values_supported	Array JSON contenente un elenco degli algoritmi di firma ("valori *alg*") supportati dal *token endpoint* per la firma sul JWT utilizzato per autenticare il client al *token endpoint*. Vedi **RFC 8414#section-2**.
require_signed_ request_object	Booleano. DEVE essere impostato a *true* per indicare che la richiesta di autorizzazione è protetta usando un Request Object firmato [**RFC 9101**].
request_object_ signing_alg_values_ supported	Array JSON contenente un elenco degli algoritmi di firma ("valori *alg*") supportati per i *Request Objects*. Vedi [openid-connect-discovery-1_0].
dpop_signing_alg_ values_supported	Array JSON contenente un elenco degli algoritmi di firma (valori "*alg*") supportati per i JWT DPoP proof. Vedi **RFC 9449**.
jwks	JSON Web Key Set contenente le chiavi crittografiche per '*authorization server*. Vedi OID-FED Sezione 5.2.1 e JWK.

Importante

Se `token_endpoint_auth_methods_supported` include `attest_jwt_client_auth`, l'Authorization Server DEVE includere entrambi `client_attestation_signing_alg_values_supported` e `client_attestation_pop_signing_alg_values_supported` nei propri metadati. I client DOVREBBERO recuperare e analizzare i metadati per rilevare supporto e requisiti di algoritmo per l'Attestation-Based Client Authentication e, in caso di incompatibilità, POSSONO ottenere una nuova attestation con un algoritmo supportato.

Metadata per openid_credential_issuer

I Metadata *openid_credential_issuer* contiene i seguenti *claims*.

Claim	Descrizione
credential_issuer	OBBLIGATORIO. L'identificativo del Fornitore di Attestati Elettronici. DEVE essere un HTTPS URL *case sensitive* come definito in OpenID4VCI Sezioni 12.2.1 e 12.2.4.
credential_endpoint	OBBLIGATORIO. URL del *Credential endpoint*. Vedi OpenID4VCI Sezione 12.2.4.
nonce_endpoint	OBBLIGATORIO. URL del *Nonce endpoint*, come definito nella Sezione 7 di OpenID4VCI.
deferred_ credential_endpoint	OPZIONALE. URL del *deferred credential endpoint*, come definito nella Sezione 12.2.4 di OpenID4VCI.
notification_ endpoint	OBBLIGATORIO. DEVE essere un URL HTTPS che indica il *notification endpoint*. Vedi Sezione 12.2.4 di [OpenID4VCI].
authorization_ servers	OPZIONALE. Array di stringhe, dove ogni stringa è un identificativo dell'*authorization server* OAuth 2.0 (come definito in [**RFC 8414**]) usato dal Fornitore di Attestati Elettronici per gestire l'autenticazione/autorizzazione. Se questo parametro è omesso vuol dire che il Fornitore di Attestati Elettronici agisce direttamente anche come *authorization server*.
display	OBBLIGATORIO. Vedi OpenID4VCI Sezione 12.2.4 Array di oggetti contenenti proprietà di visualizzazione della lingua. I parametri che DEVONO essere inclusi sono: • **name**: Denominazione in formato stringa del Fornitore di Attestati Elettronici. • **locale**: Valore stringa che identifica la localizzazione rappresentato come un tag linguistico come definito in *BCP47* **RFC 5646**. DEVE esserci un solo oggetto per ogni identificativo di localizzazione. • **logo**: Oggetto contenente informazioni relative al logo del Credential Issuer. Contiene i seguenti parametri: – **uri**: OBBLIGATORIO. URL del logo dell'entità che verrà mostrato all'Utente durante le interazioni con l'istanza del Wallet. Il MIME type del logo DEVE essere `application/svg`. – **uri#integrity**: OPZIONALE. "integrity metadata" come definito nella Sezione 3 del documento W3C-SRI. – **alt_text**: OPZIONALE. Stringa contenente il testo da mostrare in alternativa all'immagine del logo.

continues on next page

Tabella 10.7 – continua dalla pagina precedente

Claim	Descrizione
credential_ configurations_ supported	OBBLIGATORIO. Oggetto JSON che delinea i dettagli dell'Attestato Elettronico supportato dal Fornitore di Attestato Elettronico. Include un elenco di coppie nome/valore, dove ogni nome identifica in modo univoco un specific Attestato Elettronico supportato. Questo identificativo viene utilizzato per informare l'Istanza del Wallet su quale Attestato Elettronico può essere emesso. Il valore associato all'interno dell'oggetto DEVE contenere Metadata specifici per quell'Attestato Elettronico, come definito di seguito. Vedi OpenID4VCI Sezioni 12.2.4 e A.3.2. • **format**: Stringa che identifica il formato di questo Attestato Elettronico. L'Attestato Elettronico DEVE supportare il valore stringa "*dc+sd-jwt*" nel caso di SD-JWT VC (Vedi OpenID4VCI Sezione A.3. 1.) e "*mso_mdoc*" nel caso di mdoc (vedi OpenID4VCI Sezione A.2.1.). • **scope**: Stringa JSON che identifica il valore *scope* supportato. L'Istanza del Wallet DEVE utilizzare questo valore nella *Pushed Authorization Request* inviata. I valori di scope DEVONO essere l'intero insieme o un sottoinsieme dei valori *scope* presenti nel parametro *scopes_supported* del *authorization server*. Se l'Attestato Elettronico è incluso nel Catalogo degli Attestati Elettronici, il valore scope DEVE corrispondere al parametro `credential_type` definito in *Struttura del Catalogo degli Attestati Elettronici* (pagina 206) oppure in *Registro degli Schema* (pagina 232). [Vedi OpenID4VCI Sezione 12.2.4]. • **cryptographic_binding_methods_supported**: Array JSON di stringhe *case sensitive* che identificano la rappresentazione della chiave crittografica di *binding* dell'Attestato Elettronico emesso. Il Fornitore di Attestato Elettronico DEVE supportare il valore "*jwk*" per il formato "dc+sd-jwt" e "*cose_key*" per "mso_mdoc". • **credential_signing_alg_values_supported**: Array JSON di stringhe *case sensitive* che identificano gli algoritmi che il Fornitore di Attestato Elettronico DEVE supportare per firmare l'Attestato Elettronico emesso. Vedi Sezione *Algoritmi Crittografici* (pagina 647) per maggiori dettagli. • **proof_types_supported**: Oggetto JSON che fornisce informazioni dettagliate sulle *key proof* supportate dal Fornitore di Attestato Elettronico. Consiste in un elenco di coppie nome/valore, dove ogni nome identifica in modo univoco il *proof type* supportato. Il Fornitore di Attestato Elettronico DEVE supportare almeno "*jwt*" come definito in OpenID4VCI Appendice F.1. Il valore associato a ciascuna coppia nome/valore è un oggetto JSON contenente informazioni relative alla *key proof(s)*. Il Fornitore di Attestato Elettronico DEVE supportare almeno il parametro **proof_signing_ alg_values_supported** che DEVE essere un Array JSON di stringhe *case sensitive* che identificano gli algoritmi supportati (vedi Sezione *Algoritmi Crittografici* (pagina 647) per maggiori dettagli sugli algoritmi supportati). Il Fornitore di Attestato Elettronico PUO supportare il parametro **key_attestations_required** come definito nella Sezione 12.2 di OpenID4VCI. • **vct**: OBBLIGATORIO solo se `format` è valorizzato con "*dc+sd-*

Tabella 10.7 – continua dalla pagina precedente

Claim	Descrizione
jwks	OBBLIGATORIO. JSON Web Key Set, passato per valore, contenente le chiavi specifiche del protocollo usato dal Fornitore di Attestato Elettronico. Vedi OID-FED Sezione 5.2.1 e JWK.
trust_frameworks_supported	**OBBLIGATORIO. Array JSON contenente tutti i trust framework supportati. I valori supportati sono:** • *it_cie*: trust framework CIE id supportato. • *it_wallet*: trust framework IT-Wallet supportato. • *eudi_wallet*: trust framework Member State EUDI Wallet supportato. • *it_l2+document_proof*: protocollo Autenticazione eID Substantial con Verifica MRTD supportato.
batch_credential_issuance	OPZIONALE. Oggetto contenente informazioni sull'emissione di Credenziali in batch da parte del Credential Issuer presso il Credential Endpoint. La presenza di questo parametro indica che il Credential Issuer supporta più di una prova di possesso nel parametro `proofs` nella Credential Request, pertanto può emettere più di un Attestato Elettronico con gli stessi attributi relativi al titolare in un'unica richiesta/risposta. Il parametro che DEVE essere incluso è: • **batch_size**: Valore intero che specifica la dimensione massima dell'array per il parametro `proofs` nella Credential Request. La presenza di questo parametro dichiara l'emissione in batch solo per i tipi di Attestato per i quali il Credential Issuer documenta il supporto. Il Credential Issuer DEVE dichiarare se PID e IT-Wallet ID rientrano nel perimetro dell'emissione in batch. Un tipo di Attestato senza tale dichiarazione DEVE essere emesso come singolo Attestato (*CI_203* (pagina 825)).
status_list_aggregation_endpoint	OBBLIGATORIO. URL del *Status List Aggregation Endpoint*. Vedi TOKEN-STATUS-LIST Sezione 9.

10.3 Soluzione di Relying Party

Una Relying Party, come entità Organizzativa che si affida all'IT-Wallet, fornisce Soluzioni Tecniche (Soluzione di Relying Party) che potrebbero combinare software, hardware, servizi, impostazioni e configurazioni, incluse App di Verifica per l'autenticazione dell'Utente e la verifica degli Attestati Elettronici.

Per implementare e fornire Soluzioni Tecniche, una Relying Party può avvalersi dei servizi di un **Intermediario di Relying Party**. Ai sensi dell'articolo 5b, paragrafo 8, del EU_2024_1183, gli intermediari che agiscono per conto delle Relying Party sono considerati Relying Party e **non devono conservare dati sul contenuto della transazione** tra l'Utente del Wallet e la Relying Party intermediata. Per le definizioni complete e il quadro giuridico europeo di riferimento, vedi **Intermediario di Relying Party** e **Intermediario IT-Wallet** in *Definizioni e Acronimi* (pagina 675).

Nota

Nel profilo implementativo descritto dalle presenti specifiche tecniche, un Intermediario di Relying Party è anche un Intermediario di Federazione (**Intermediario IT-Wallet**).

Una Relying Party fornisce almeno uno dei seguenti componenti:

- **Backend della Relying Party**: Gestisce la registrazione delle Istanze di Relying Party e la relativa gestione dei Certificati. Ottiene il Certificato X.509 secondo l'*L'Infrastruttura di Trust* (pagina 117). DOVREBBE inoltre fornire un Certificato X.509 alle proprie App di Verifica. Può anche implementare servizi web aggiuntivi e logica di business per i propri scopi e casi d'uso.
- **App di Verifica**: È un frontend fornito agli Utenti per accedere ai Servizi della Relying Party, e può essere fornito come:
  - **Client Web**.
  - **Applicazione Mobile**.

Il seguente diagramma illustra l'Architettura ad alto livello della Soluzione di Relying Party.

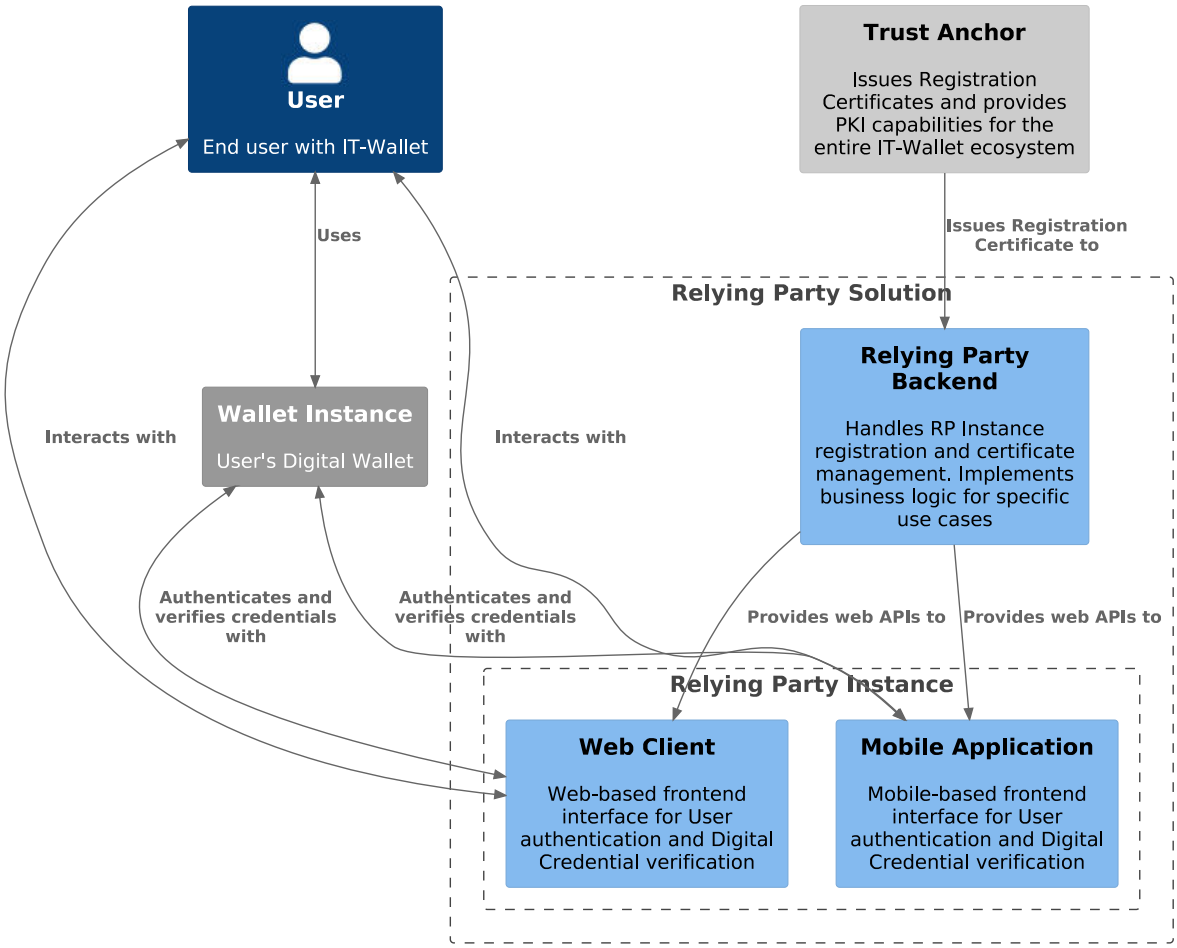


Fig. 10.11: Architettura di Alto Livello della Soluzione di Relying Party.

10.3.1 App di Verifica

L'App di Verifica è un'applicazione mobile o embedded progettata per richiedere, ricevere ed elaborare Attestati Elettronici dalle Istanze del Wallet in modo affidabile. Ogni App di Verifica garantisce l'integrità, la riservatezza e l'autenticità degli scambi di Attestati Elettronici, consentendo interazioni sicure tra Utenti e Relying Party.

Esistono due tipi principali di App di Verifica, ciascuna destinata a diversi ambienti operativi:

- **App di Verifica Embedded**: una soluzione hardware/software che opera su un dispositivo specializzato (ad es., varchi di accesso). Ogni App di Verifica corrisponde a un'installazione specifica dell'applicazione su un dispositivo embedded. Poiché la trust con l'App di Verifica Embedded è stabilita attraverso la trust instaurata con il Mobile Relying Party Provider, l'App di Verifica Embedded può essere considerata un Confidential Client di OAuth.
- **App di Verifica Mobile**: un'applicazione nativa che opera su un dispositivo mobile (ad es., smartphone o tablet). Ogni istanza corrisponde a un'installazione specifica dell'applicazione su un dispositivo. Poiché la trust con la App di Verifica Mobile è stabilita attraverso la trust instaurata con il Mobile Relying Party Provider, l'App di Verifica Mobile può essere considerata un Confidential Client di OAuth.
- **App di Verifica Web**: un'applicazione remota gestita dalla Relying Party. Un'App di Verifica Web opera come Confidential Client di OAuth, il che significa che può memorizzare in modo sicuro configurazioni riservate (come le sue chiavi crittografiche private) su un server remoto. In questo contesto, la Relying Party non è fornita da terzi e la richiesta di presentazione e la successiva validazione sono gestite automaticamente dal software.

Nota

Le App di Verifica Embedded e Mobile che agiscono come client pubblici non sono considerate all'interno di queste specifiche.

Nota

A differenza della App di Verifica Web, un'App di Verifica Mobile o Embedded richiedono una gestione appropriata del ciclo di vita e procedure ad hoc di registrazione gestite attraverso il Backend della Relying Party.

Ulteriori dettagli tecnici e operativi sono discussi nelle sezioni seguenti.

10.3.2 App di Verifica Mobile

Il ciclo di vita di un'App di Verifica include quattro stati principali: **Installed**, **Unverified**, **Verified** e **Uninstalled**, supportando funzionalità come registrazione, riemissione del Certificato di Accesso e revoca.

Ciclo di Vita App di Verifica Mobile

In questa sezione vengono presentati le macchine a stati finiti per spiegare gli stati dell'App di Verifica Mobile, nonché le loro transizioni e relazioni.

Come mostrato in Fig. 10.12, l'App di Verifica Mobile ha quattro stati distinti: **Installed**, **Unverified**, **Verified** e **Uninstalled**. Ogni stato rappresenta uno stato funzionale specifico e determina le azioni che possono essere eseguite.

Transizione a Installed

La macchina a stati inizia con l'installazione della App di Verifica (transizione **RPI INST**), dove gli Utenti scaricano e installano una App di Verifica utilizzando l'app store ufficiale del sistema operativo del loro dispositivo, portando allo stato **Installed**.

In questo stato, l'App di Verifica DEVE interagire solo con il Backend della Relying Party per essere registrata (cioè, per verificare l'integrità dell'Istanza, registrare le Cryptographic Hardware Keys e ottenere un Certificato di Accesso).

Quando avviene la revoca dell'App di Verifica (transizione **RPI REV**), l'App di Verifica DEVE tornare da **Verified** a **Installed**. Questa transizione implica le seguenti operazioni:

1. Il Certificato di Accesso DEVE essere revocato.
2. Le Cryptographic Hardware Keys DEVONO essere cancellate.

La revoca può avvenire nei seguenti casi:

- Per motivi di sicurezza (ad es., compromissione del materiale crittografico).
- Per motivi tecnici (ad es., deprecazione della Soluzione di Relying Party).
- In caso di de-registrazione della Relying Party (come dettagliato in EIDAS-ARF, Sezione 6.4.3).
- Attività illegali segnalate da Organi Giudiziari o di Supervisione.

Inoltre, ogni Relying Party DOVREBBE stabilire un periodo di tempo (periodo di grazia) durante il quale l'App di Verifica può richiedere presentazioni di Credenziali Elettroniche autenticandosi verso un'Istanza del Wallet utilizzando un Certificato di Accesso scaduto. Dopo questo periodo, l'App di Verifica DEVE essere de-registrata (transizione **RPI DEREG**) e

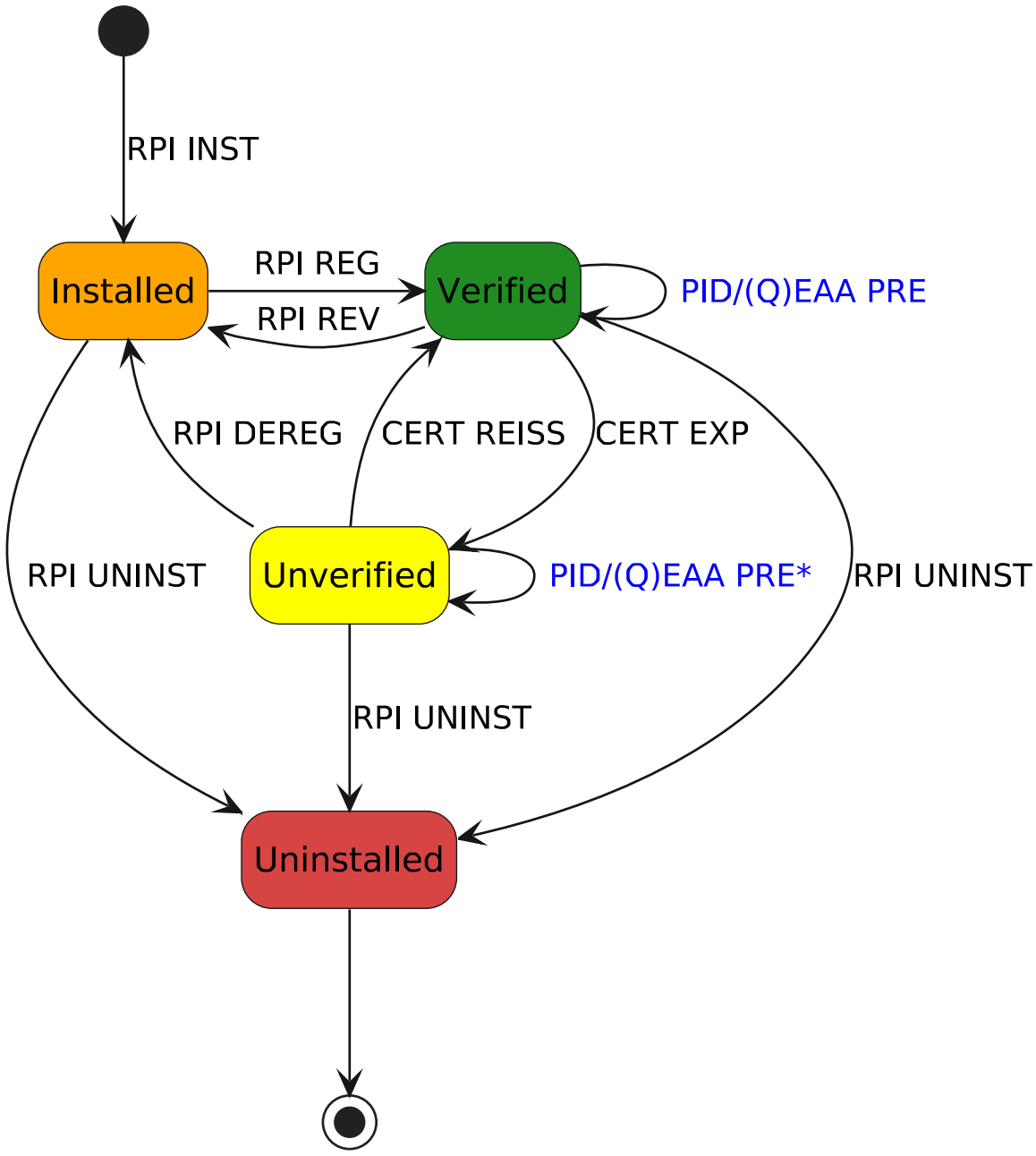


Fig. 10.12: Ciclo di vita della App di Verifica Mobile.

tornare allo stato **Installed**. Questa transizione implica che le Cryptographic Hardware Keys DEVONO essere cancellate.

Transizione a Verified

L'App di Verifica deve ottenere un Certificato di Accesso appropriato, che sarà utilizzato per autenticarsi verso le Istanze del Wallet. Questo Certificato viene ottenuto interagendo con il Backend della Relying Party, che a sua volta comunica con l'Autorità di Certificazione per i Certificati di Accesso dell'App di Verifica. In particolare, la transizione di registrazione (**RPI REG**) consiste nelle seguenti sottofasi, che portano allo stato **Verified**:

1. **Inizializzazione**: Dopo la verifica dell'integrità dell'App di Verifica, questa registra una coppia di Cryptographic Hardware Keys.
2. **Emissione del Certificato di Accesso**: L'App di Verifica ottiene un Certificato di Accesso.

Nel caso in cui il Certificato di Accesso sia scaduto, un nuovo Certificato di Accesso può essere emesso per l'App di Verifica; questa operazione è rappresentata dalla transizione **CERT REISS** verso lo stato **Verified**.

In questo stato, l'App di Verifica può richiedere la presentazione di Attestati Elettronici alle Istanze del Wallet (**PID/(Q)EAA PRE**), utilizzando il Certificato di Accesso per autenticarsi.

Transizione a Unverified

La scadenza del Certificato di Accesso (transizione **CERT EXP**) porta allo stato **Unverified**.

In questo stato, l'App di Verifica può ancora richiedere la presentazione di Attestati Elettronici alle Istanze del Wallet durante il periodo di grazia. Tuttavia, poiché il Certificato è scaduto, un disclaimer specifico DEVE essere mostrato all'Utente dell'Istanza del Wallet durante il flusso di presentazione; per questo motivo, questa operazione è rappresentata dall'etichetta **PID/(Q)EAA PRE**. Questo è necessario per supportare i flussi di presentazione offline. Dopo che il periodo di grazia è trascorso, la Relying Party Instance NON DEVE più richiedere presentazioni e sarà de-registrata.

Transizione a Uninstalled

Attraverso gli stati **Installed**, **Verified** e **Unverified**, la Relying Party Instance può essere rimossa completamente attraverso la disinstallazione dell'App di Verifica (transizione **RPI UNINST**), portando allo stato **Uninstalled**. Se un'App di Verifica è **Uninstalled**, termina il suo ciclo di vita.

Funzionalità dell'App di Verifica Mobile

Un'App di Verifica DEVE supportare tre funzionalità fondamentali: **Registrazione**, **Riemissione del Certificato di Accesso** e **Revoca**. Ogni funzionalità è descritta in dettaglio nelle sezioni seguenti.

Nota

In questa sezione, i servizi utilizzati per attestare l'autenticità dell'App di Verifica e del dispositivo in cui è installata sono indicati come **Servizio di Integrità del Dispositivo**. Il Servizio di Integrità del Dispositivo è considerato in modo astratto e si presume che sia un servizio fornito da una terza parte affidabile (cioè, l'API del fornitore del sistema operativo) in grado di eseguire controlli di integrità sull'Istanza del Wallet e sul dispositivo in cui è installata.

Nota

I dettagli forniti di seguito sono non normativi e sono destinati a chiarire le funzionalità dell'App di Verifica Mobile. L'implementazione effettiva può variare in base al caso d'uso specifico e ai requisiti della Relying Party.

Registrazione App di Verifica Mobile

Questo processo consente la registrazione di un'App di Verifica con il Backend della Relying Party e l'emissione di un Certificato di Accesso che sarà utilizzato per autenticarsi verso le Istanze del Wallet durante i flussi di presentazione. Il processo consiste in due sottofasi:

1. **Inizializzazione**: L'App di Verifica richiede un'asserzione di sicurezza e integrità dal produttore del sistema operativo, che lega anche una coppia di chiavi crittografiche asimmetriche a lunga durata memorizzata in un adeguato storage sicuro all'interno del dispositivo stesso; l'asserzione viene quindi convalidata dal Backend della Relying Party. Ulteriori dettagli sono forniti nella Sezione *Istanza dell'Applicazione Mobile* (pagina 705).

2. **Emissione del Certificato di Accesso**: L'App di Verifica richiede un Certificato di Accesso dal Backend della Relying Party. Prima di interagire con l'Autorità di Certificazione per l'emissione del Certificato di Accesso, il Backend della Relying Party convalida l'integrità e la sicurezza dell'App di Verifica sfruttando le chiavi attestate a lunga durata generate nella sottofase precedente. Il flusso è mostrato in Fig. 10.13, mentre una descrizione passo-passo è fornita di seguito.

Nota

I Certificati di Accesso POSSONO essere emessi come a breve durata (tipicamente validi entro 24 ore) o a lunga durata.

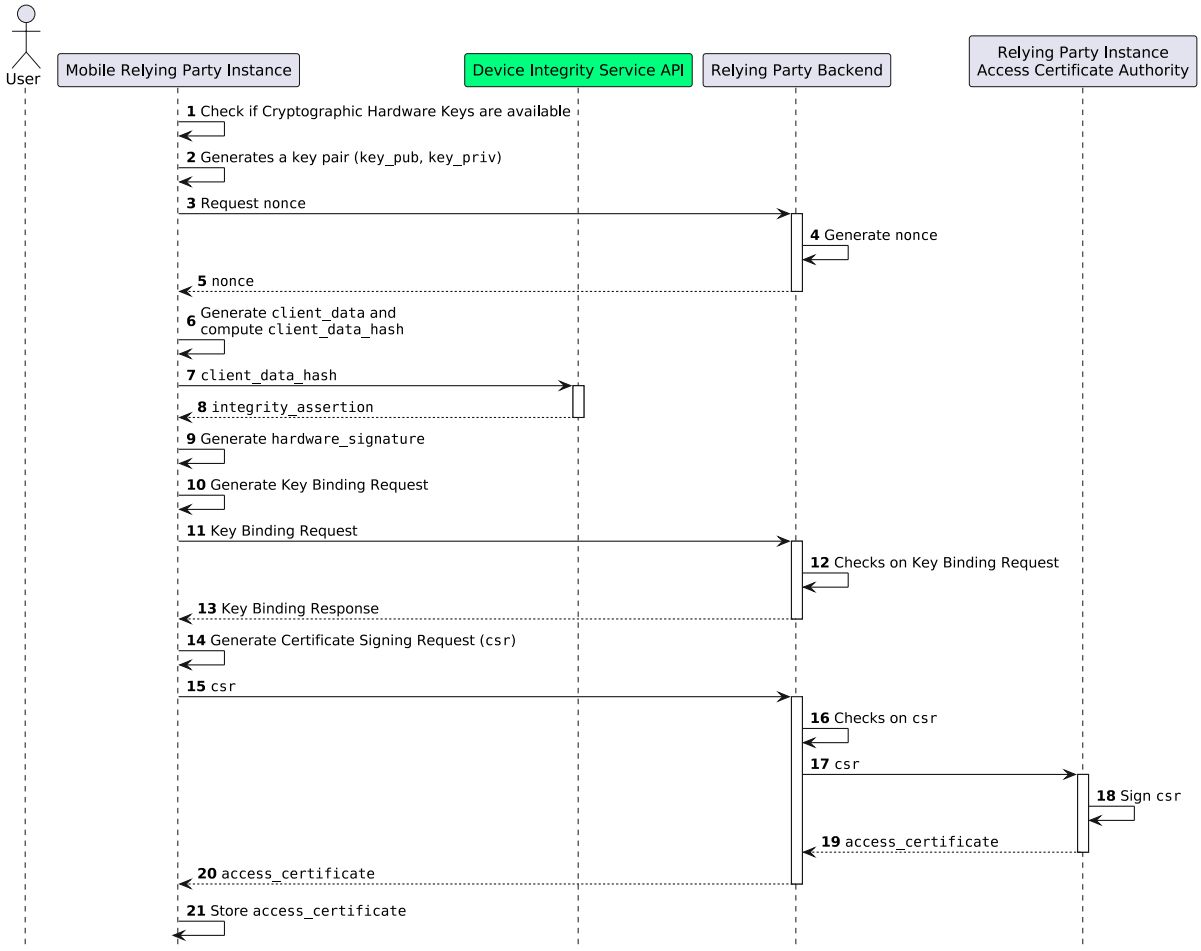


Fig. 10.13: Flusso di Emissione del Certificato di Accesso della App di Verifica Mobile.

Passi 1-2: La App di Verifica Mobile:

1. Verifica l'esistenza delle Cryptographic Hardware Keys. Se non esistono, è necessaria la re-inizializzazione dell'App di Verifica.

2. Genera una coppia di chiavi asimmetriche per il Certificato di Accesso (`key_pub`, `key_priv`).

Passi 3-5: L'App di Verifica Mobile richiede un `nonce` dall'relying-party-provider-backend-endpoint:Endpoint Nonce della Relying Party del Backend della Relying Party. Questo `nonce` DEVE essere imprevedibile per servire come principale difesa contro i replay attack.

In caso di richiesta riuscita, l'relying-party-provider-backend-endpoint:Endpoint Nonce della Relying Party genera e restituisce il `nonce` all'App di Verifica Mobile. L'relying-party-provider-backend-endpoint:Endpoint Nonce della Relying Party DEVE garantire che sia monouso e valido solo entro un periodo di tempo specifico.

Esempi non normativi della Richiesta e Risposta del Nonce possono essere trovati rispettivamente nelle sezioni *Richiesta di Nonce dell'Applicazione Mobile* (pagina 710) e *Risposta di Nonce dell'Applicazione Mobile* (pagina 711).

Passo 6: L'App di Verifica:

1. Genera `client_data`, un JSON Object che include `nonce` e l'impronta digitale di `key_pub`, ottenuta dalla sua rappresentazione `JWK`.
2. Calcola `client_data_hash` applicando l'algoritmo SHA256 a `client_data`.

Di seguito è riportato un esempio non normativo di `client_data`.

```
{
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
  "jwk_thumbprint": "hT3v7KQjFZy6GvDkYgOZ1u2F6T4Nz5bPjX8o1MZ3dJY"
}
```

Passi 7-8: L'App di Verifica Mobile:

1. Richiede al Servizio di Integrità del Dispositivo di creare un valore `integrity_assertion` collegato al `client_data_hash`.
2. Riceve un valore `integrity_assertion` firmato dal Servizio di Integrità del Dispositivo, autenticato dall'OEM.

Passi 9-11: L'App di Verifica Mobile:

1. Genera un valore `hardware_signature` firmando il `client_data_hash` con la chiave privata Cryptographic Hardware, che servirà come prova di possesso per le Cryptographic Hardware Keys.
2. Genera la relying-party-provider-backend-endpoint:Richiesta di Associazione Chiavi della Relying Party sotto forma di JWT. Questo JWT include `integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag` e `cnf` (che rappresenta `key_pub`); è firmato utilizzando `key_priv`.
3. Invia il JWT firmato relying-party-provider-backend-endpoint:Richiesta di Associazione Chiavi della Relying Party come parametro `assertion` nel corpo di HTTP request all'relying-party-provider-backend-endpoint:Endpoint di Associazione Chiavi della Relying Party.

Passo 12: Il Backend della Relying Party valuta la Richiesta di Key Binding ed esegue i seguenti controlli:

1. La richiesta include tutti i parametri di intestazione HTTP richiesti come definito in relying-party-provider-backend-endpoint:Richiesta di Associazione Chiavi della Relying Party.
2. La firma della Richiesta di Key Binding è valida e verificabile utilizzando il `jwk` fornito.
3. Il valore `nonce` è stato generato dal Backend della Relying Party e non è stato utilizzato in precedenza.
4. L'App di Verifica possiede valide Cryptographic Hardware Keys registrate.
5. Il `client_data` può essere ricostruito utilizzando `nonce` e `cnf` (che rappresenta `key_pub`). Il valore del parametro `hardware_signature` viene quindi convalidato utilizzando la chiave pubblica della Cryptographic Hardware Key registrata associata all'App di Verifica.
6. L'`integrity_assertion` può essere convalidata secondo le linee guida del produttore del dispositivo. I controlli specifici eseguiti dal Backend della Relying Party sono dettagliati nella documentazione del produttore del sistema operativo.
7. Il dispositivo in uso è privo di difetti di sicurezza noti e soddisfa i requisiti minimi di sicurezza definiti dalla Relying Party.
8. L'URL nel parametro `iss` corrisponde all'identificatore URL della Relying Party.

Passo 13: Se i controlli hanno successo, il Backend della Relying Party risponde con una conferma di successo (relying-party-provider-backend-endpoint:Risposta di Associazione Chiavi della Relying Party).

Passo 14: L'App di Verifica Mobile genera una Certificate Signing Request (CSR, `csr`) utilizzando `key_pub` e `key_priv`.

Passo 15: L'App di Verifica Mobile invia la CSR al relying-party-provider-backend-endpoint: Endpoint del Certificato di Accesso della Relying Party del Backend della Relying Party, come parte della relying-party-provider-backend-endpoint:Richiesta del Certificato di Accesso della Relying Party.

Passi 16-17: Il Backend della Relying Party verifica che la chiave pubblica nella CSR corrisponda a un'App di Verifica che è stata precedentemente convalidata, cioè che corrisponda a quella legata alle Cryptographic Hardware Keys attraverso `hardware_signature`. Se questo controllo ha successo, il Backend della Relying Party invia la CSR all'Autorità di Certificazione per i Certificati di Accesso dell'App di Verifica.

Passi 18-19: L'Autorità di Certificazione per i Certificati di Accesso dell'App di Verifica firma la CSR, ottenendo un Certificato di Accesso valido (`access_certificate`) che invia al Backend della Relying Party.

Passi 20-21: Il Backend della Relying Party invia il Certificato di Accesso (come parte della relying-party-provider-backend-endpoint:Risposta del Certificato di Accesso della Relying Party) all'App di Verifica Mobile, che lo memorizza per future autenticazioni verso le Istanze del Wallet.

Riemissione de Certificato di Accesso App di Verifica Mobile

L'emissione di un nuovo Certificato di Accesso segue lo stesso flusso descritto nella sezione *Registrazione App di Verifica Mobile* (pagina 367) per **Emissione del Certificato di Accesso**. Questi certificati POSSONO essere emessi come a breve durata (tipicamente validi entro 24 ore) o a lunga durata.

Revoca App di Verifica Mobile

Le Relying Party DEVONO verificare periodicamente l'autenticità e la sicurezza delle App di Verifica. Quando vengono rilevati problemi di sicurezza, le Relying Party DEVONO revocare l'App di Verifica, revocando il suo Certificato di Accesso X.509 (in caso di certificati a lunga durata), e in ogni caso, le Relying Party NON DEVONO consentire la riemissione di certificati. Di conseguenza, la revoca dell'App di Verifica Mobile DEVE essere legata alla validità dei Certificati di Accesso X.509.

I Certificati X.509 a lunga durata seguono i requisiti relativi al loro ciclo di vita, definiti in *L'Infrastruttura di Trust* (pagina 117).

10.3.3 App di Verifica Web

Le Web Instance operano controlli di sicurezza lato server che memorizzano in modo sicuro segreti e chiavi crittografiche in un ambiente controllato. Le App di Verifica Web DEVONO essere registrate presso la Trust Anchor o le Entità Intermediarie, secondo *L'Infrastruttura di Trust* (pagina 117).

Funzionalità dell'App di Verifica Web

Un'App di Verifica Web DEVE supportare due funzionalità fondamentali: **Registrazione** e **Revoca**. Ogni funzionalità è descritta nelle sezioni seguenti.

Registrazione Revoca App di Verifica Web

Le App di Verifica Web, in qualità di OAuth Confidential Client, sono registrate direttamente dalla Trust Anchor o un'Entità Intermediaria. La registrazione comporta:

- La Relying Party DEVE registrare la sua App Web con la Trust Anchor o con le appropriate Entità Intermediarie della Federazione.
- La Relying Party DEVE esporre una Entity Configuration come definito nel Trust Framework.
- L'Entity Configuration DEVE contenere tutti i metadati necessari per la federazione, inclusi gli endpoint e le chiavi pubbliche.
- Non è richiesta alcuna gestione del ciclo di vita delle singole istanze, poiché l'App Web opera come parte dell'ambiente server protetto.

Revoca App di Verifica Web

Quando una App di Verifica Web deve essere revocata:

- La revoca DEVE essere eseguita secondo le procedure del Trust Framework.
- Le chiavi crittografiche utilizzate dall'App Web DEVONO essere revocate.
- L'Entity Configuration DEVE essere aggiornata per riflettere la revoca.
- La Trust Anchor DEVE essere notificata della revoca per aggiornare i metadata della federazione.

10.3.4 Entity Configuration Relying Party

Secondo la Sezione *Configurazione della Federazione* (pagina 125), come Entità di Federazione, la Relying Party è tenuta a mantenere un endpoint well-known che ospita la sua Entity Configuration. L'Entity Configuration delle Relying Party DEVE contenere i parametri definiti nelle Sezioni *Entity Configuration Foglie e Intermediari* (pagina 130) e *Parametri Comuni delle Entity Configuration* (pagina 127).

Le Relying Party DEVONO fornire i seguenti tipi di metadata:

- `federation_entity`
- `openid_credential_verifier`

I metadata contenuti in `federation_entity` DEVONO contenere i claim come definito nella Sezione *Metadati di federation_entity Foglie* (pagina 131).

Entity Configuration di una Relying Party

Di seguito un esempio non normativo della richiesta effettuata dall'Istanza del Wallet all'endpoint well-known `openid-federation` per ottenere l'Entity Configuration della Relying Party:

```
GET /.well-known/openid-federation HTTP/1.1
HOST: relying-party.example.org
```

Di seguito un esempio non normativo di risposta:


```
{
    "iat": 1718207217,
    "exp": 1749743216,
    "iss": "https://relying-party.example.org",
    "sub": "https://relying-party.example.org",
    "authority_hints": [
        "https://trust-anchor.example.org"
    ],
    "jwks": {
        "keys": [
            {
                "kid": "FANFS3YnC9tjiCaivhWLVUJ3AxwGGz_98uRFaqMEEs",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
            }
        ]
    },
    "metadata": {
        "federation_entity": {
            "homepage_uri": "https://relying-party.example.org",
            "organization_name": "Organization Name",
            "contacts": [
                "informazioni@example.it",
                "protocollo@pec.example.it"
            ],
            "tos_uri": "https://relying-party.example.org/public/info_policy.html",
            "policy_uri": "https://relying-party.example.org/public/privacy_policy.html",
            "logo_uri": "https://relying-party.example.org/public/logo.svg"
        },
        "openid_credential_verifier": {
            "application_type": "web",
            "client_id": "https://relying-party.example.org",
            "client_name": "Organization Name",
            "logo_uri": "https://relying-party.example.org/public/compact-logo.svg",
            "contacts": [
```

```
        "informazioni@example.it",
        "protocollo@pec.example.it"
    ],
    "request_uris": [
        "https://relying-party.example.org/request_uri"
    ],
    "response_uris": [
        "https://relying-party.example.org/response_uri"
    ],
    "redirect_uris": [
        "https://relying-party.example.org/cb"
    ],
    "encrypted_response_enc_values_supported": [
        "A256GCM"
    ],
    "vp_formats_supported": {
        "dc+sd-jwt": {
            "sd-jwt_alg_values": [
                "ES256",
                "ES384",
                "ES512"
            ],
            "kb-jwt_alg_values": [
                "ES256"
            ]
        },
        "mso_mdoc": {
            "issuerauth_alg_values": [
                -9,
                -51
            ],
            "deviceauth_alg_values": [
                -9,
                -51
            ]
        }
    },
    "jwks": {
        "keys": [
            {
                "kid": "f10aca0992694b3581f6f699bfc8a2c6cc687725",
                "kty": "EC",
                "crv": "P-256",
                "x": "jE2RpcQbFQxKpMqehahgZv6smmXD0i/LTP2QRzMADk4",
                "y": "qkMx5iqt5PhPu5tfctS6HsP+FmLgrxfrzUV2GwMQuh8"
            }
        ]
```

```
            }
        }
    }
}
```

10.3.5 Metadati della Relying Party

I metadata *openid_credential_verifier* DEVONO contenere il *client_metadata*, come incluso nei parametri mostrati di seguito. (test-plans-remote-presentation:Matrice di Test per il Verificatore di Credenziali in Remoto). Si noti che *openid_credential_verifier* è un metadata specifico di OpenID Federation considerato per OpenID4VP quando il `client_id` della Relying Party è impostato con `openid_federation`. Quando il parametro `client_id` della Relying Party è impostato con `x509_hash`, i metadata vengono invece trasmessi nel parametro `client_metadata` fornito all'interno della richiesta.

Claim	Descrizione
client_id	DEVE contenere un URL HTTPS che identifica in modo univoco la RP. Vedi **RFC 7591#section-3.2.1** e OpenID Connect Dynamic Client Registration 1.0 Sezione 3.2.
client_name	Stringa leggibile che rappresenta il nome della RP. Vedi **RFC 7591#section-2**.
logo_uri	URL del logo dell'entità che verrà mostrato all'Utente durante le interazioni con l'istanza del Wallet. Vedi OID-FED Sezione 5.2.2. Il MIME type del logo DEVE essere `application/svg`.
application_type	Stringa che indica il tipo di applicazione. DEVE essere impostata al valore "*web*". Vedi OpenID Connect Dynamic Client Registration 1.0 Sezione 2.
request_uris	JSON Array di valori `request_uri` che sono pre-registrati dalla RP. Questi URL DEVONO utilizzare lo schema *https*. Vedi OpenID Connect Dynamic Client Registration 1.0 Sezione 2.
response_uris	JSON Array di stringhe URI di risposta a cui l'Istanza del Wallet DEVE inviare la Risposta di Autorizzazione utilizzando una richiesta HTTP POST come definito dalle Modalità di Risposta `direct_post` e `direct_post.jwt` (vedi OpenID4VP Draft 20 Sezioni 6.2 e 6.3).
encrypted_response_enc_values_supported	Array JSON degli algoritmi di cifratura dei contenuti che il Verifier supporta per cifrare la risposta di autorizzazione quando si utilizza la modalità di risposta `direct_post.jwt`. Vedi OpenID4VP §5.1 e §8.3.1.

continues on next page

Tabella 10.8 – continua dalla pagina precedente

Claim	Descrizione
vp_formats_supported	Oggetto JSON che definisce i formati e i tipi di prova delle Verifiable Presentations e delle Verifiable Credentials supportati dalla RP. È composto da una lista di coppie nome/valore, dove ogni nome identifica in modo univoco un tipo supportato. La RP DEVE supportare almeno `dc+sd-jwt`. Per le SD-JWT VC, il valore associato a ciascuna coppia nome/valore DEVE includere `sd-jwt_alg_values` che elenca gli algoritmi di firma accettati; per mdoc-CBOR, la valore DEVE includere `issuerauth_alg_values` e `deviceauth_alg_values`. Gli header JOSE/COSE degli artefatti presentati DEVONO corrispondere a uno dei valori dichiarati. Vedi OpenID4VP Sezione 11 e Appendice B.
jwks	Documento JSON Web Key Set, passato per valore, contenente le chiavi specifiche del protocollo per la Relying Party. Vedi OID-FED Sezione 5.2. 1 e JWK.
erasure_endpoint	[CONDIZIONALE] Stringa JSON che rappresenta l'URI a cui l'Istanza del Wallet può richiedere la cancellazione degli attributi degli Utenti. Questo URL DEVE utilizzare lo schema `https`. Alla ricezione di una richiesta di cancellazione, la Relying Party DEVE individuare univocamente uno o più Attestati Elettronici per i quali l'Utente richiede la rimozione, applicando l'identity matching come specificato in *Identity Matching e Identity Reconciliation* (pagina 567).

Nota

I parametri `response_uris` e `erasure_endpoint` sono introdotti in questa specifica.

10.4 Fonti Autentiche

Le Fonti Autentiche forniscono gli Attributi dell'Utente ai Fornitori di Attestati Elettronici consentendo loro il rilascio degli Attestati Elettronici. Durante il Flusso di Emissione, i Fornitori di Attestati Elettronici richiedono alle Fonti Autentiche gli attributi necessari per fornire l'Attestato Elettronico richiesto dall'Utente. Le Fonti Autentiche POSSONO anche fornire una Credential Offer legata ai loro Fornitori di Attestati Elettronici come definito nella Sezione *Flusso Credential Offer* (pagina 477).

Le Fonti Autentiche Pubbliche DEVONO interagire con i Fornitori di Attestati Elettronici tramite PDND secondo le regole definite nella Sezione *e-Service PDND* (pagina 714) e nella Sezione *Aggiornamento dello Stato da parte delle Fonti Autentiche* (pagina 432). Vedere anche la Sezione *Catalogo degli e-Service PDND delle Fonti Autentiche* (pagina 634) per ulteriori dettagli.

Le Fonti Autentiche DEVONO:

- fornire gli Attributi dell'Utente quando richiesto dal Fornitore di Attestati Elettronici autorizzato a emettere il relativo Attestato Elettronico che attesta gli Attributi. Le Fonti Autentiche Pubbliche DEVONO utilizzare PDND per inviare gli Attributi dell'Utente ai loro Fornitori di Attestati Elettronici. Quando gli Attributi dell'Utente non sono disponibili durante il Flusso di Emissione, le Fonti Autentiche DEVONO fornire ai Fornitori di Attestati Elettronici un tempo stimato di quando i dati dell'Utente saranno disponibili. Le Fonti Autentiche POSSONO richiedere una prova che:
  - la richiesta degli Attributi dell'Utente sia relativa a dati che li riguardano;
  - la richiesta degli Attributi dell'Utente provenga da un'Istanza del Wallet valida;
- cooperare con i Fornitori di Attestati Elettronici in modo che gli Attributi attestati in un Attestato Elettronico siano sempre mantenuti aggiornati. Le Fonti Autentiche Pubbliche DEVONO utilizzare PDND Signal Hub per notificare ai loro Fornitori di Attestati Elettronici qualsiasi aggiornamento riguardante attributi che sono cambiati o non sono più validi.

CAPITOLO 11

Gestione degli Attestati Elettronici

11.1 Modello di Dati degli Attestati Elettronici

Un modello di dati dell'Attestato Elettronico ha la seguente struttura:

- **Attributi di metadati**:
  - **Format-Agnostic**: Sono attributi di metadati di alto livello che descrivono l'Attestato Elettronico indipendentemente dal suo formato di codifica. Rappresentano le informazioni semantiche sulla credenziale (ad esempio, credential_type, issuing_authority, expiry_date) e rimangono concettualmente coerenti tra i diversi formati. Quando una credenziale viene codificata, questi attributi di metadati comuni vengono mappati a parametri tecnici specifici del formato secondo le regole di codifica di ciascun formato (SD-JWT-VC o mdoc-CBOR).
  - **Format-Specific**: Sono parametri di metadati intrinseci al formato di codifica specifico e servono a scopi tecnici relativi al modello di sicurezza e ai requisiti di protocollo del formato.
- **Attributi dell'Utente**: Informazioni sull'Utente, come identità o qualifiche.

Gli Attestati Elettronici di Attributi (Qualificati) ((Q)EAA) sono rilasciati dai Fornitori di (Q)EAA a un'Istanza del Wallet e DEVONO essere forniti in formato SD-JWT VC o mdoc-CBOR. Mentre il modello dati (Q)EAA è guidato dal caso d'uso e può includere diversi attributi dell'Utente, gli attributi di metadati specifici per ciascun formato dati sono forniti nelle sezioni seguenti.

11.1.1 Attributi di Metadati Format-Agnostic dell'Attestato Elettronico

La seguente tabella definisce gli attributi di metadati comuni che sono applicabili agli Attestati Elettronici indipendentemente dal loro formato di codifica. Questi attributi rappresentano le informazioni semantiche sulla credenziale.

Identificativo del dato	Descrizione
credential_type_ identifier	OBBLIGATORIO. Identificatore unico e resistente alle collisioni che specifica il tipo e lo schema dell'Attestato Elettronico. Definisce l'insieme di claim/ attributi che l'Attestato Elettronico contiene e la loro struttura.
issuing_authority	OBBLIGATORIO. Nome dell'autorità amministrativa che ha emesso l'Attestato Elettronico.
issuing_country	OBBLIGATORIO. Codice paese Alpha-2, come specificato in ISO 3166-1, del paese o territorio del Fornitore di Attestati Elettronici.
issuance_date	OPZIONALE. Data (e se possibile ora) in cui l'Attestato Elettronico è stato emesso e/o in cui è iniziato il periodo di validità amministrativa dell'Attestato Elettronico.
expiry_date	OPZIONALE. Data (e se possibile ora) di scadenza dell'Attestato Elettronico.
location_status	OPZIONALE. La posizione delle informazioni sullo stato di validità dell'Attestato Elettronico dove il Fornitore di Attestati Elettronici revoca l'Attestato Elettronico.
cryptographic_ binding	OPZIONALE. Oggetto contenente il materiale crittografico per la prova di possesso.
verification	OPZIONALE. Oggetto contenente informazioni di Identity proofing e verifica dei dati dell'Utente.

Le sezioni seguenti forniscono gli attributi specifici del formato e una mappatura degli attributi di metadati sopra indicati ai parametri tecnici specifici del formato quando la credenziale è codificata in formato SD-JWT VC o mdoc-CBOR.

11.1.2 Formato Attestato Elettronico SD-JWT-VC

Quando gli Attestati Elettronici sono emessi in formato SD-JWT-VC, DEVONO essere conformi alle specifiche SD-JWT e SD-JWT-VC.

SD-JWT DEVE essere firmato utilizzando la chiave privata del Fornitore di Attestati Elettronici. SD-JWT DEVE essere fornito insieme a un Type Metadata relativo all'Attestato Elettronico emesso secondo le Sezioni 6 e 6.3 di [SD-JWT-VC]. Il payload DEVE contenere il claim **__sd_ alg** descritto nella Sezione 4.1.1 SD-JWT e gli altri claim specificati in questa sezione.

Il claim **__sd_alg** indica l'algoritmo di hash utilizzato dal Fornitore di Attestati Elettronici per generare i digest come descritto nella Sezione 4.1.1 di SD-JWT. **__sd_alg** DEVE essere impostato su uno degli algoritmi specificati nella Sezione *Cryptographic Algorithms* (pagina 647).

I claim che non sono divulgabili selettivamente DEVONO essere inclusi nell'SD-JWT così come sono. I digest delle disclosure, insieme a eventuali decoy se presenti, DEVONO essere contenuti nell'array **_sd**, come specificato nella Sezione 4.2.4.1 di SD-JWT.

Le Disclosure vengono fornite al Titolare insieme all'SD-JWT nel *Combined Format for Issuance* che è una serie ordinata di valori codificati in base64url, ciascuno separato dal successivo da un singolo carattere tilde ('~') come segue:

```
<Issuer-Signed-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure N>~
```

Vedere SD-JWT-VC e SD-JWT per ulteriori dettagli.

Attributi Metadata degli Attestati Elettronici SD-JWT

Il JOSE Header contiene i seguenti parametri obbligatori:

Claim	Descrizione	Riferimento
typ	OBBLIGATORIO. DEVE essere valorizzato con `dc+sd-jwt` come definito in SD-JWT-VC. NON DEVE essere valorizzato con `none`.	**RFC 7515** Sezione 4.1.9.
alg	OBBLIGATORIO. Algoritmo di firma.	**RFC 7515** Sezione 4.1.1.
kid	OBBLIGATORIO. Identificativo univoco della chiave pubblica.	**RFC 7515** Sezione 4.1.8.
trust_chain	OPZIONALE. Array JSON contenente la catena di fiducia che dimostra l'affidabilità di chi emette il JWT.	[OID-FED] Sezione 4.3.
x5c	OBBLIGATORIO. Contiene il certificato della chiave pubblica X.509 o la catena di certificati [**RFC 5280**] corrispondente alla chiave utilizzata per firmare digitalmente il JWT.	**RFC 7515** Sezione 4.1.8 e [SD-JWT-VC] Sezione 3.5.

Il payload JWT contiene i seguenti claim. Salvo diversamente specificato, i seguenti claim NON DEVONO essere divulgabili selettivamente.

Claim	Descrizione	Riferimento
iss	OBBLIGATORIO. *Stringa*. Stringa URL che rappresenta l'identificativo univoco del Fornitore di Attestati Elettronici.	[RFC7519, Sezione 4.1.1].

continues on next page

Tabella 11.3 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
sub	OPZIONALE. *Stringa.* L'identificativo del soggetto dell'Attestato Elettronico, l'Utente, DEVE essere un valore opaco e NON DEVE corrispondere a nessun dato anagrafico o essere derivato dai dati anagrafici dell'Utente tramite pseudonimizzazione. Inoltre, è richiesto che due diverse Credenziali emesse NON DEVONO utilizzare lo stesso valore di `sub`.	[RFC7519, Sezione 4.1.2].
iat	OPZIONALE. Timestamp UNIX con l'orario di emissione del JWT, codificato come NumericDate come indicato in **RFC 7519**.	[RFC7519, Sezione 4.1.6].
exp	OBBLIGATORIO. Timestamp UNIX con l'orario di scadenza del JWT, codificato come NumericDate come indicato in **RFC 7519**. In conformità al requisito [EIDAS-ARF] **ISSU_12c** e **ISSU_12d** il timestamp NON DEVE essere successivo alla data di scadenza della Wallet Unit Attestation presentata durante il processo di emissione della Credenziale Digitale.	[RFC7519, Sezione 4.1.4].
nbf	OPZIONALE. Timestamp UNIX con l'orario di inizio validità del JWT, codificato come NumericDate come indicato in **RFC 7519**.	[RFC7519, Sezione 4.1.4].
issuing_authority	OBBLIGATORIO. *Stringa.* Identificativo del dato *issuing_authority* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380).	Regolamento di esecuzione della Commissione EU_2024/2977.
issuing_country	OBBLIGATORIO. *Stringa.* Identificativo del dato *issuing_country* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380).	Regolamento di esecuzione della Commissione EU_2024/2977.
issuance_date	OPZIONALE. *Stringa.* Identificativo del dato *issuance_date* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Questo attributo si riferisce alla data amministrativa di emissione dell'Attestato Elettronico, che è tipicamente diverso dalla data tecnica di emissione espresso dal claim JWT `iat`.	Sezione 2.6 dell'ARF PID Rulebook v1.3 [EIDAS-ARF].
date_of_expiry	OPZIONALE. *Stringa.* Identificativo del dato *expiry_date* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Questo attributo si riferisce al periodo di validità amministrativa dell'Attestato Elettronico, che è tipicamente diverso dal periodo di validità tecnica espresso dal claim JWT `exp`.	Regolamento di esecuzione della Commissione EU_2024/2977.

continues on next page

Tabella 11.3 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
status	OPZIONALE. OBBLIGATORIO solo se l'Attestato Elettronico è long-lived. *Oggetto JSON*. Identificativo del dato *location_status* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). DEVE contenere il membro JSON *status_list*.	Sezione 3.2.2.2 SD-JWT-VC.
cnf	OPZIONALE. *Oggetto JSON*. Identificativo del dato *cryptographic_binding* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380), contenente il materiale crittografico per la prova di possesso. Includendo un claim **cnf** (confirmation) in un JWT, il Fornitore del JWT dichiara che il Titolare ha il controllo della chiave privata relativa a quella pubblica definita nel parametro **cnf**. Il destinatario DEVE verificare crittograficamente che il Titolare abbia effettivamente il controllo di quella chiave.	[RFC7800, Sezione 3.1] e Sezione 3.2.2.2 SD-JWT-VC.
vct	OBBLIGATORIO. *Stringa*. Identificativo del dato *credential_type_identifier* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Il valore del tipo di Attestato Elettronico DEVE essere una URN e DEVE essere impostato utilizzando uno dei valori ottenuti dai metadata del Fornitore di Attestati Elettronici, il confronto dei caratteri letterali inclusi in questa URN DEVE essere eseguito in modo case-sensitive. È l'identificativo del tipo di SD-JWT VC e DEVE essere impostato con un valore resistente alle collisioni come definito nella Sezione 2 di **RFC 7515**. DEVE contenere anche il numero di versione del tipo di Attestato Elettronico. A meno che non sia diversamente specificato da EIDAS-ARF e dagli EUDI Rulebook, il `vct` DOVREBBE seguire una struttura come `urn:it-wallet:{credential_type}:{credential_type_version}`.	Sezione 3.2.2.2 SD-JWT-VC.
vct#integrity	OPZIONALE. *Stringa*. Il valore DEVE essere una stringa "integrity metadata" come definito nella Sezione 3 di [W3C-SRI]. *SHA-256*, *SHA-384* e *SHA-512* DEVONO essere supportati come funzioni crittografiche di hash. *MD5* e *SHA-1* NON DEVONO essere utilizzati. Questo claim DEVE essere verificato in base a quanto indicato nella Sezione 3.3.5 di [W3C-SRI].	Sezione 6.1 SD-JWT-VC, [W3C-SRI]

continues on next page

Tabella 11.3 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
verification	OPZIONALE. *Oggetto JSON*. Identificativo del dato *verification* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Include i seguenti sotto-valori: • `trust_framework`: OBBLIGATORIO. *Stringa* che identifica il trust framework utilizzato per l'autenticazione dell'Utente. DEVE essere impostato utilizzando uno dei valori descritti nella mappa *trust_frameworks_supported* fornita nei Metadata del Fornitore di Attestati Elettronici. • `assurance_level`: OBBLIGATORIO. *Stringa* che identifica il livello di garanzia dell'identità garantito durante il processo di autenticazione dell'Utente. Il valore DEVE corrispondere a uno dei valori mappati nell'array `acr_values_supported` presente nel credential-issuer-metadata:Metadata del Fornitore di Attestati Elettronici.	Estensione domestica.
_sd	OBBLIGATORIO. *Array di stringhe*, dove ogni stringa rappresenta un digest di una Disclosure.	4.2.4.1 SD-JWT
_sd_alg	OBBLIGATORIO. *Stringa*. Algoritmo di hash utilizzato dal Fornitore di Attestati Elettronici per generare i digest.	4.1.1 SD-JWT

Nota

I claim JWT standard `nbf` e `exp` sono utilizzati per esprimere il periodo di validità tecnica di un Attestato Elettronico conforme a SD-JWT VC.

Il parametro `status_list` di `status` DEVE essere un Oggetto JSON conforme alla Sezione 6.2 di TOKEN-STATUS-LIST.

Esempi Non Normativi di (Q)EAA

Di seguito è riportato un esempio non normativo di (Q)EAA in JSON.

```
{
    "iss": "https://issuer.example.org",
    "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
    "iat": 1683000000,
    "exp": 1883000000,
    "issuing_authority": "Credential Issuer Organization",
```

```
    "issuing_country": "IT",
    "date_of_expiry": "2033-03-19",
    "status": {
        "status_list": {
            "idx": 5678,
            "uri": "https://issuer.example.org/status"
        }
    },
    "vct": "urn:it-wallet:disabilitycard:1",
    "vct#integrity": "2e40bcd6799008085ffb1a1f3517efee335298fd976b3e655bfb3f4eaa11d171",
    "verification": {
        "trust_framework": "it_wallet",
        "assurance_level": "https://trust-anchor.example.it/loa/high"
    },
    "document_number": "XXXXXXXXXX",
    "given_name": "Mario",
    "family_name": "Rossi",
    "birth_date": "1980-01-10",
    "tax_id_code": "TINIT-XXXXXXXXXXXXXXXX",
    "constant_attendance_allowance": true
}
```


Il corrispondente SD-JWT è rappresentato di seguito, con header e payload decodificati in JSON.


```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
    "_sd": [
        "D4VkWjnA0WON7HdCGFtU869MSvORHPf8p5fQRD5gNj0",
        "JOQk0kuBSVk80rFlv9VGY-yiIzsfzEJKk3d4RROfzkM",
        "Q7TX7kL8CNUp3BFBKP5xxIuPu5gRgkO6HplM3E1iMIc",
        "Wq3gFfmC0I9Lefw1mh-Bk5XPRtoSCg9aE23uOhxakas",
        "oF2qeWAbKO_qWGQ5z-HGKeifl2PMIEMbJe8L-PJ-wko",
        "qbRtUHp9Oax9dm5GeKnw_W12Yu1E2DoU6wrFPee7aBo"
    ],
    "exp": 1883000000,
    "iss": "https://issuer.example.org",
    "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
    "iat": 1683000000,
    "issuing_authority": "Credential Issuer Organization",
    "issuing_country": "IT",
```

```
    "date_of_expiry": "2033-03-19",
    "status": {
        "status_list": {
            "idx": 5678,
            "uri": "https://issuer.example.org/status"
        }
    },
    "vct": "urn:it-wallet:disabilitycard:1",
    "vct#integrity": "2e40bcd6799008085ffb1a1f3517efee335298fd976b3e655bfb3f4eaa11d171",
    "verification": {
        "trust_framework": "it_wallet",
        "assurance_level": "https://trust-anchor.example.it/loa/high"
    },
    "_sd_alg": "sha-256",
    "cnf": {
        "jwk": {
            "kty": "EC",
            "crv": "P-256",
            "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
            "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
        }
    }
}
```


Di seguito è riportato l'elenco delle disclosure:

Claim `document_number`:

- Hash SHA-256: `D4VkWjnA0WON7HdCGFtU869MSvORHPf8p5fQRD5gNj0`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImRvY3VtZW50X251bWJlciIsICJYWFhYWFhYWFhYIl0`
- Contenuto: `["kghte5MDNHbQfdJHp88pCA", "document_number", "XXXXXXXXXX"]`

Claim `given_name`:

- Hash SHA-256: `qbRtUHp9Oax9dm5GeKnw_W12Yu1E2DoU6wrFPee7aBo`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contenuto: `["hX1TEz_z877_XAtr3COaWg", "given_name", "Mario"]`

Claim `family_name`:

- Hash SHA-256: `Q7TX7kL8CNUp3BFBKP5xxIuPu5gRgkO6HplM3E1iMIc`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contenuto: `["YWtI06xDdCyvTalcInTE3A", "family_name", "Rossi"]`

Claim `birth_date`:

- Hash SHA-256: `oF2qeWAbKO_qWGQ5z-HGKeifl2PMIEMbJe8L-PJ-wko`
- Disclosure: `WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgImJpcnRoX2RhdGUiLCAiMTk4MC0wMS0xMCJd`
- Contenuto: `["-z34cJ1gC5UBPCIx8OhNiQ", "birth_date", "1980-01-10"]`

Claim `tax_id_code`:

- Hash SHA-256: `Wq3gFfmC0I9Lefw1mh-Bk5XPRtoSCg9aE23uOhxakas`
- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgInRheF9pZF9jb2RlIiwgIlRJTklULVhYWFhYWFhYWFhYWFgiXQ`
- Contenuto: `["KNc5-Gk9CQh_TdGbqBKI7A", "tax_id_code", "TINIT-XXXXXXXXXXXXXXXX"]`

Claim `constant_attendance_allowance`:

- Hash SHA-256: `JOQk0kuBSVk80rFlv9VGY-yiIzsfzEJKk3d4RROfzkM`
- Disclosure: `WyIyaFFtWXBIeVgtbVpKaHoyeHNVWWNRIiwgImNvbnN0YW50X2F0dGVuZGFuY2VfYWxsb3dhbmNlIiwgdHJ1ZV0`
- Contenuto: `["2hQmYpHyX-mZJhz2xsUYcQ", "constant_attendance_allowance", true]`

Il formato combinato per l'emissione del (Q)EAA è rappresentato di seguito:


```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2N2dM
N2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMiOiBbIjxJ
c3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19.eyJfc2QiOlsiRDRWa1dqbkEwV09ON0h
kQ0dGdFU4NjlNU3ZPUkhQZjhwNWZRUkQ1Z05qMCIsIkpPUWswa3VCU1ZrODByRmx2OVZ
HWS15aUl6c2Z6RUpLazNkNFJST2Z6a00iLCJRN1RYN2tMOENOVXAzQkZCS1A1eHhJdVB
1NWdSZ2tPNkhwbE0zRTFpTUljIiwiV3EzZ0ZmbUMwSTlMZWZ3MW1oLUJrNVhQUnRvU0N
nOWFFMjN1T2h4YWthcyIsIm9GMnFlV0FiS09fcVdHUTV6LUhHS2VpZmwyUE1JRU1iSmU
4TC1QSi13a28iLCJxYlJ0VUhwOU9heDlkbTVHZUtud19XMTJZdTFFMkRvVTZ3ckZQZWU
3YUJvIl0sImV4cCI6MTg4MzAwMDAwMCwiaXNzIjoiaHR0cHM6Ly9pc3N1ZXIuZXhhbXB
sZS5vcmciLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25vV1hGWkFmSGt4WnNSR0M5WHMiLCJ
pYXQiOjE2ODMwMDAwMDAsImlzc3VpbmdfYXV0aG9yaXR5IjoiQ3JlZGVudGlhbCBJc3N
1ZXIgT3JnYW5pemF0aW9uIiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V
4cGlyeSI6IjIwMzMtMDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4Ijo
1Njc4LCJ1cmkiOiJodHRwczovL2lzc3Vlci5leGFtcGxlLm9yZy9zdGF0dXMifX0sInZ
jdCI6InVybjppdC13YWxsZXQ6ZGlzYWJpbGl0eWNhcmQ6MSIsInZjdCNpbnRlZ3JpdHk
iOiIyZTQwYmNkNjc5OTAwODA4NWZmYjFhMWYzNTE3ZWZlZTMzNTI5OGZkOTc2YjNlNjU
1YmZiM2Y0ZWFhMTFkMTcxIiwidmVyaWZpY2F0aW9uIjp7InRydXN0X2ZyYW1ld29yayI
6Iml0X3dhbGxldCIsImFzc3VyYW5jZV9sZXZlbCI6Imh0dHBzOi8vdHJ1c3QtYW5jaG9
yLmV4YW1wbGUuaXQvbG9hL2hpZ2gifSwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOns
iandrIjp7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoiVENBRVIxOVp2dTNPSEY
0ajRXNHZmU1ZvSElQMUlMaWxEbHM3dkNlR2VtYyIsInkiOiJaeGppV1diWk1RR0hWV0t
WUTRoYlNJaXJzVmZ1ZWNDRTZ0NGpUOUYySFpRIn19fQ.rU0-nlNtkaqNS1h8h23YMs1m
BUIgJQDY6Ouq__Ih0gBEqviTrtkHZu_MgfLWRbAwnu59ubHo0PKkuy3c6D19SA~WyJrZ
2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImRvY3VtZW50X251bWJlciIsICJYWFhYWFhYW
```

```
FhYIl0~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImdpdmVuX25hbWUiLCAiTWFya
W8iXQ~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImZhbWlseV9uYW1lIiwgIlJvc3
NpIl0~WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgImJpcnRoX2RhdGUiLCAiMTk4MC
0wMS0xMCJd~WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgInRheF9pZF9jb2RlIiwgI
lRJTklULVhYWFhYWFhYWFhYWFgiXQ~WyIyaFFtWXBIeVgtbVpKaHoyeHNVWWNRIi
wgImNvbnN0YW50X2F0dGVuZGFuY2VfYWxsb3dhbmNlIiwgdHJ1ZV0~
```

Type Metadata dell'Attestato Elettronico

Il documento di *Type Metadata*, se fornito, DEVE essere un *JSON object* e DEVE essere conforme con la Sezione 6.2 di [SD-JWT-VC].

In aggiunta ai parametri definiti nella sezione 9.2 di [SD-JWT-VC], il seguente parametro DOVREBBE essere incluso:

- `name`: Nome "human-readable" dell'Attributo destinato agli utenti finali.

In conformità con la Sezione 6.3.3 di SD-JWT-VC, il documento JSON del *Type Metadata* PUÒ essere recuperato tramite un *well-known* endpoint. Questo endpoint, fornito dal Fornitore di Attestati Elettronici, DEVE avere il seguente formato: `https://{Dominio Credential Issuer}/.well-known/type-metadata`. A tale endpoint DEVE essere aggiunto il parametro di query `vct`. L'endpoint restituisce un codice di stato `200 OK` e supporta `application/json` come content type.

Di seguito è riportato un esempio non normativo.

```
GET /.well-known/type-metadata?vct=urn%3Aeudi%3Apid%3Ait%3A1 HTTP/1.1
Host: issuer.example.it
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json

{
  "vct": "urn:eudi:pid:it:1",
  "name": "...",
  "description": "...",
  ...
}
```

11.1.3 Formato Attestato Elettronico mdoc-CBOR

Gli Attestati Elettronici emessi in formato mdoc-CBOR DEVONO essere basati sullo standard ISO/IEC 18013-5.

I dati in mdoc DEVONO essere codificati in CBOR come definito in **RFC 8949**.

Questo modello dati struttura gli Attestati Elettronici in componenti distinti: namespaces (**name Spaces**) e prova crittografica (**issuerAuth**). I namespace categorizzano e strutturano i dati (o attributi, vedi *Attributi dei Namespaces* (pagina 392)). Mentre la prova crittografica garantisce integrità e autenticità attraverso il Mobile Security Object (MSO).

L'MSO memorizza in modo sicuro i digest crittografici degli attributi all'interno dei *nameSpaces*. Ciò consente alle Relying Party di convalidare gli attributi divulgati rispetto ai valori **digest ID** corrispondenti senza rivelare l'intero Attestato Elettronico. Vedere *Mobile Security Object* (pagina 390) per i dettagli.

Un Attestato Elettronico in formato mdoc-CBOR DEVE avere la seguente struttura:

Parametro	Descrizione	Riferimento
nameSpaces	*(map)*. I namespace all'interno dei quali sono definiti i dati. Un Attestato Elettronico PUÒ includere più namespace.	[ISO 18013-5#8.3.2.1.2]
issuerAuth	*(COSE_Sign1)*. Contiene il *Mobile Security Object* (MSO), un documento COSE Sign1, emesso dal Fornitore di Attestati Elettronici.	[ISO 18013-5#9.1.2.4]

Nota

Gli attributi mDL obbligatori utilizzano il namespace standard *org.iso.18013.5.1*. Tuttavia, PUÒ avere anche un namespace domestico, come *org.iso.18013.5.1.IT*, per includere attributi aggiuntivi definiti in questo profilo di implementazione. Ogni namespace all'interno di *name Spaces* DEVE condividere lo stesso valore del tipo di documento emesso (*docType*), che identifica la natura dell'Attestato Elettronico, come definito in *issuerAuth*.

La struttura di una Credenziale in formato mdoc-CBOR è ulteriormente descritta nelle sezioni seguenti.

Mobile Security Object

L'**issuerAuth** rappresenta il *Mobile Security Object* che è un *Documento COSE Sign1* definito in **RFC 9052**. Ha la seguente struttura di dati:

- protected header
- unprotected header
- payload
- signature

Il **protected header** DEVE contenere il seguente parametro codificato in formato CBOR:

Elemento	Descrizione	Riferimento
1	*(int)*. Algoritmo utilizzato per verificare la firma crittografica dell'Attestato Elettronico in formato mdoc.	**RFC 9053**

Nota

Solo l'algoritmo di firma DEVE essere presente nel protected header, altri elementi NON DOVREBBERO essere presenti.

L'**unprotected header** DEVE contenere i seguenti parametri, se non diversamente specificato:

Elemento	Descrizione	Riferimento
4	*(tstr, OPZIONALE)*. Identificativo univoco del JWK dell'Emittente. Richiesto quando l'Emittente del documento mdoc utilizza OpenID Federation.	*L'Infrastruttura di Trust* (pagina 117)
33	*(array)*. Catena di certificati X.509 relativa all'Emittente. Obbligatorio se l'autenticazione è basata su certificato X.509.	**RFC 9360**

Nota

x5chain è incluso nell'unprotected header con lo scopo di consentire al Titolare di aggiornare la catena di certificati X.509, relativa all'emittente del *Mobile Security Object*, senza invalidare la firma.

Il **payload** DEVE contenere il *MobileSecurityObject*, senza il parametro di header COSE Sign *content-type* e codificato come una *byte string* (bstr) utilizzando il *CBOR Tag* 24.

Il *MobileSecurityObject* DEVE avere i seguenti attributi, se non diversamente specificato:

Elemento	Descrizione	Riferimento
docType	*(tstr)*. Codifica formato dell'Identificativo del dato *credential_type_identifier* come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). • Se definito da uno standard ISO, DEVE essere una stringa della forma `iso.org.{iso-number}.{part}.{version}.{credential_type}` (per esempio, per una mDL, il valore DEVE essere `org.iso.18013.5.1.mDL`). • Se definito a livello europeo, DEVE essere una stringa della forma `eu.europa.ec.eudi.{credential_type}.{version}` (ad es., `eu.europa.ec.eudi.pid.1`). • Se definito a livello nazionale, DEVE essere una stringa della forma `{Trust Anchor reverse domain}.{credential_type}.{version}` (ad es., `it-wallet.trust-registry.pid.1`).	[ISO 18013-5#9.1.2.4]
version	*(tstr)*. Versione del *MobileSecurityObject*.	[ISO 18013-5#9.1.2.4]
validityInfo	*(map, OBBLIGATORIO)*. Contiene le date e gli orari di emissione e scadenza del *MobileSecurityObject*. Include i seguenti sub-valori: • **signed** *(tdate, OPZIONALE)*. Il timestamp che indica quando il *MobileSecurityObject* è stato firmato. • **validFrom** *(tdate, OPZIONALE)*. Timestamp prima del quale il *MobileSecurityObject* non è considerato valido. Quando presente, DEVE essere uguale o successivo a *signed*. • **validUntil** *(tdate, OBBLIGATORIO)*. Timestamp dopo il quale il *MobileSecurityObject* non è più considerato valido. In conformità al requisito [EIDAS-ARF] **ISSU_12c** e **ISSU_12d** il timestamp NON DEVE essere successivo alla data di scadenza della Wallet Unit Attestation presentata durante il processo di emissione della Credenziale Digitale.	[ISO 18013-5#9.1.2.4]
digestAlgorithm	*(tstr)*. Identificativo dell'algoritmo di digest, che DEVE corrispondere all'algoritmo definito nel protected header.	[ISO 18013-5#9.1.2.4]
valueDigests	*(map)*. Associa ogni namespace a un insieme di digest, dove ogni digest è indicizzato da un *digestID* univoco e contiene il valore del digest.	[ISO 18013-5#9.1.2.4]

continues on next page

Tabella 11.7 – continua dalla pagina precedente

Elemento	Descrizione	Riferimento
deviceKeyInfo	*(map)*. Contiene le informazioni relative alla chiave pubblica dell'Istanza del Wallet. DEVE includere i seguenti sub parametri, se non diversamente specificato: • **deviceKey** *(COSE_Key)*. Contiene i parametri relativi alla chiave pubblica. • **keyAuthorizations** *(map, OPZIONALE)*. Definisce le autorizzazioni per gli interi namespaces o per i singoli dati. • **keyInfo** *(map, OPZIONALE)*. Contiene metadati aggiuntivi della chiave.	[ISO 18013-5#9.1.2.4]
status	*(map, OPZIONALE)*. OBBLIGATORIO solo se l'Attestato Elettronico ha durata maggiore di 24 ore (long-lived). Identificativo del dato *location_status* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Contiene le informazioni relative allo stato di revoca del MSO. Se presente, include una *status_list* basata sul meccanismo definito nella Sezione6.3 di TOKEN-STATUS-LIST.	[ISO 18013-5#9.1.2.6]

Nota

La chiave privata relativa alla chiave pubblica memorizzata nel *deviceKey* viene utilizzata per firmare i *DeviceSignedItems* e per dimostrare il possesso dell'Attestato Elettronico durante la fase di presentazione (vedere la fase di presentazione con mdoc-CBOR).

Attributi dei Namespaces

nameSpaces contiene una o più voci *nameSpace*, ciascuna identificata da un nome. All'interno di ogni **nameSpace**, sono inclusi uno o più *IssuerSignedItemBytes*, ciascuno codificato in una stringa di byte codificata in CBOR con Tag 24 (#6.24(bstr .cbor)), che appare come 24(«... ») nella notazione diagnostica. Essa rappresenta le informazioni da divulgare, una per ogni digest presente all'interno del *Mobile Security Object* e DEVE contenere i seguenti attributi:

Nome	Descrizione	Riferimento
digestID	*(uint)*. Valore identificativo di uno dei `ValueDigests` forniti nel *Mobile Security Object*.	[ISO 18013-5#9.1.2.5]

continues on next page

Tabella 11.8 – continua dalla pagina precedente

Nome	Descrizione	Riferimento
random	*(bstr)*. Valore di byte casuale utilizzato come *salt* per la funzione di hash. Questo valore DEVE essere diverso per ogni *IssuerSignedItem* e DEVE avere una lunghezza minima di 16 byte.	[ISO 18013-5#9.1.2.5]
elementIdentifier	*(tstr)*. Nome identificativo del dato.	[ISO 18013-5#8.3.2.1.2.3]
elementValue	*(any)*. Valore del dato.	[ISO 18013-5#8.3.2.1.2.3]

Attributi Metadata degli Attestati Elettronici mdoc-CBOR

I seguenti **elementIdentifiers** che rappresentano attributi metadata format-encoded sono definiti per gli Attestati Elettronici in formato mdoc-CBOR all'interno del rispettivo *name Space*:

Element Identifier	Descrizione	Riferimento
issuing_country	*(tstr, OBBLIGATORIO)*. Identificativo del dato format-encoded *issuing_country* come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Codice paese Alpha-2 come definito in [ISO 3166-1].	[ISO 18013-5#7.2]
issuing_authority	*(tstr, OBBLIGATORIO)*. Identificativo del dato format-encoded *issuing_authority* come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Il valore DEVE contenere solo caratteri Latin1b e deve avere una lunghezza massima di 150 caratteri.	[ISO 18013-5#7.2]
issuance_date	*(tdate o full-date, OPZIONALE)*. Identificativo del dato *issuance_date* codificato nel formato come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). Questo attributo si riferisce alla data amministrativa di emissione dell'Attestato Elettronico, che è tipicamente diverso dalla data tecnica di emissione espresso dai parametri `signed` or `validFrom` del *MobileSecurityObject*.	Sezione 2.6 dell'ARF PID Rulebook v1.3 [EIDAS-ARF].
expiry_date	*(tdate o full-date, OPZIONALE)*. Identificativo del dato format-encoded *expiry_date* come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). DEVE essere conforme al formato ISO 8601-1 YYYY-MM-DD.	Sezione 3 dell'ARF PID Rulebook v1.3 [EIDAS-ARF]

continues on next page

Tabella 11.9 – continua dalla pagina precedente

Element Identifier	Descrizione	Riferimento
sub	*(uuid, OPZIONALE)*. Identifica il soggetto dell'Attestato Elettronico mdoc (l'Utente). L'identificativo DEVE essere opaco, NON DEVE corrispondere a nessun dato anagrafico e NON DEVE essere derivato dai dati anagrafici dell'Utente attraverso la pseudonimizzazione. Inoltre, Attestati Elettronici diversi emessi allo stesso Utente o a Utenti diversi NON DEVONO utilizzare lo stesso valore *sub*.	Estensione domestica.
verification	*(map, OPZIONALE)*. Identificativo del dato format-encoded *verification* come definito nella Sezione *Attributi di Metadati Format-Agnostic dell'Attestato Elettronico* (pagina 380). La CBOR map include i seguenti membri: • `trust_framework` *(tstr, OBBLIGATORIO)*: trust framework utilizzato per l'autenticazione dell'Utente. • `assurance_level` *(tstr, OBBLIGATORIO)*: livello di garanzia dell'identità garantito durante l'autenticazione dell'Utente. Il valore DEVE corrispondere a uno dei valori mappati nell'array `acr_values_supported` presente nel credential-issuer-metadata:Metadata del Fornitore di Attestati Elettronici.	Estensione domestica.

Nota

Gli attributi specifici dell'Utente dell'Attestato Elettronico sono definiti nel Catalogo degli Attestati Elettronici. Gli attributi specifici dell'Utente per gli Attestati Elettronici in formato mdoc come quelli del PID o mDL sono inclusi facendo riferimento agli corrispettivi *element Identifiers* definiti in ISO/IEC 18013-5 o nella specifica EIDAS-ARF.

Nota

Indipendentemente dal tipo di Attestato Elettronico, il valore di `sub` NON DEVE essere mostrato all'Utente, in quanto non è un attributo dello stesso. È utilizzato per scopi di identificazione dagli Emittenti di Credenziali.

Esempi mdoc-CBOR

Un esempio non normativo di un mDL codificato in CBOR è mostrato di seguito in codifica binaria.

a26a6e616d65537061636573a2716f72672e69736f2e31383031332e352e318cd818a100a468646967657374494400
6672616e646f6d5820790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a9640671656c656d65
6e744964656e7469666965726b66616d696c795f6e616d656c656c656d656e7456616c756565526f737369d818a101
a468646967657374494404016672616e646f6d58201442881e24514517333019ec24aecaa70bba927d7f2d38ad7cdc3c
e82d8561db71656c656d656e744964656e7469666965726a676976656e5f6e616d656c656c656d656e7456616c7565
654d6172696fd818a102a468646967657374494402667 2616e646f6d582051b4f3831d910861e81da746b221fd8949
8507476418cedc3709b5d28a7c41d071656c656d656e744964656e7469666965726a62697274685f646174656c656c
656d656e7456616c7565d903ec6a313938302d30312d3130d818a103a468646967657374494403667 2616e646f6d58
200c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f71656c656d656e744964656e7469
666965726a69737375655f646174656c656c656d656e7456616c7565d903ec6a323032352d30332d3237d818a104a4
6864696765737449440466726 16e646f6d58200a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e333967
0203178e71656c656d656e744964656e7469666965726b6578706972795f646174656c656c656d656e7456616c7565
d903ec6a323033302d30332d3237d818a105a468646967657374494405667 2616e646f6d58204b315ff17cf3a4754a
94a6cf1e9ddfdd99f6e86b177b74f173348968ca74e80b71656c656d656e744964656e7469666965726f6973737569
6e675f636f756e7472796c656c656d656e7456616c7565624954d818a106a468646967657374494406667 2616e646f
6d5820f8c82c4103f603435d0bc7762074ccc7c2c74925314a1fb5a8ab9cf2a960221f71656c656d656e744964656e
7469666965727169737375696e675f617574686f726974796c656c656d656e7456616c756578284973746974697475746f
20506f6c696772616669636f2065205a656363612064656c6c6f20537461746fd818a107a468646967657374494407
6672616e646f6d58201f7a77a353da7bfc4da12691185249c31d421afd59ddac34f9e4fb4d92b8ec5071656c656d65
6e744964656e7469666965726b62697274685f706c6163656c656c656d656e7456616c756564526f6d61d818a108a4
6864696765737449440866726 16e646f6d582088e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6
c97cf21471656c656d656e744964656e7469666965726f646f63756d656e745f6e756d6265726c656c656d656e7456
616c756569585831323334353637d818a109a468646967657374494409667 2616e646f6d5820944758b43602b01ad6
8911b062349845492c04c6a78129bcf8cb5fb1396af2fc71656c656d656e744964656e7469666965726 8706f727472
6169746c656c656d656e7456616c7565590412ffd8ffe000104a46494600010101009000900000ffdb004300130d0e
110e0c13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241435f8260687175
7b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f7676767676767676767676767676
7676767676767676767676767676767676767676767676767676767676767676767676767676767676ffc0001108001800640301
2200021101031101ffc4001b0000030100030100000000000000000000000005060401020307ffc40032100001030303
0205020309000000000000010203040005110612211331141551617122410781a1163542527391b2c1f1ffc4001501
0101000000000000000000000000000000001ffc4001a1101010100030100000000000000000000000014111213161ff
da000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e7709723a1d0dae146ddfb
b3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb95ff00eb9ff14d215de66af089ce44b7
dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65d6b95a147381ea0d495b933275fe6bba75c114104a8ba4
10413e983dff004f5af5d34b4b4cde632d0bf1fd1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e85600
1ebc03c7ce29dd9eef1ef10fc447dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e545287
50946ec6704850cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493f602a075
559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7290875db4be88ab384bbbb
bfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30ab420a7a6d3240c718cf031fa9ef4c9ad550
205aa02951df4a1d6c8421b015b769db8c9229837ea2be8b1b0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2
edbd4584e7af9c64e3f96b9beb28d4ac40931e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4
a179c1c1e460938f9149ef655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dc
e15eed9b71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c5256a79e706d52
d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344e847dcac9dcb5abba8d145061d
43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a15f71dd27d451453e09d1a21450961cbb3ea8a956
433b781f1ce33dfed54f0e2b50a2b71d84ed6db18028a28175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a
66b726e33de9ea6f1b52b181c760724e47b514520a5a28a283ffd9d818a10aa468646967657374494 40a6672616e64
6f6d5820577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f71656c656d656e74496465
6e7469666965727264726976696e675f70726976696c65675f6c656e656d656e7456616c7565581a3757695686963
6c655f63617465676f72795f636f646561416a69737375655f64617465d903ec4b6a323032302d30392d31376b6578
706972795f64617465d903ec4b6a323033312d30362d3130d818a10ba468646967657374494 40b6672616e646f6d58
20fa21d3d890af5f4ea2760d08fd9a6256004cd5aa9d5e697ba5873fb0cddd555e71656c656d656e744964656e7469
6669657276756e5f646973746964e6775697368696e675f7369676e6c656c656d656e7456616c756561497467 2672e

La Notazione Diagnostica del mDL codificato in CBOR è riportata di seguito.

```
{
"nameSpaces": {
  "org.iso.18013.5.1": [
    24(<<
        {
        "digestID": 0,
        "random": h'790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a96406',
        "elementIdentifier": "family_name",
        "elementValue": "Rossi"
        }
      >>),
    24(<<
        {
        "digestID": 1,
        "random": h'1442881e24514517333019ec24aecaa70bba927d7f2d38ad7cdc3ce82d8561db',
        "elementIdentifier": "given_name",
        "elementValue": "Mario"
        }
      >>),
    24(<<
        {
        "digestID": 2,
        "random": h'51b4f3831d910861e81da746b221fd89498507476418cedc3709b5d28a7c41d0',
        "elementIdentifier": "birth_date",
        "elementValue": 1004(1980-01-10)
        }
      >>),
    24(<<
        {
        "digestID": 3,
        "random": h'0c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f',
        "elementIdentifier": "issue_date",
        "elementValue": 1004(2025-03-27)
        }
      >>),
    24(<<
        {
        "digestID": 4,
        "random": h'0a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e3339670203178e',
        "elementIdentifier": "expiry_date",
        "elementValue": 1004(2030-03-27)
        }
      >>),
    24(<<
        {
```

```
    "digestID": 5,
    "random": h'4b315ff17cf3a4754a94a6cf1e9ddfdd99f6e86b177b74f173348968ca74e80b',
    "elementIdentifier": "issuing_country",
    "elementValue": "IT"
    }
  >>),
24(<<
    {
    "digestID": 6,
    "random": h'f8c82c4103f603435d0bc7762074ccc7c2c74925314a1fb5a8ab9cf2a960221f',
    "elementIdentifier": "issuing_authority",
    "elementValue": "Istituto Poligrafico e Zecca dello Stato"
    }
  >>),
24(<<
    {
    "digestID": 7,
    "random": h'1f7a77a353da7bfc4da12691185249c31d421afd59ddac34f9e4fb4d92b8ec50',
    "elementIdentifier": "birth_place",
    "elementValue": "Roma"
    }
  >>),
24(<<
    {
    "digestID": 8,
    "random": h'88e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6c97cf214',
    "elementIdentifier": "document_number",
    "elementValue": "XX1234567"
    }
  >>),
24(<<
    {
    "digestID": 9,
    "random": h'944758b43602b01ad68911b062349845492c04c6a78129bcf8cb5fb1396af2fc',
    "elementIdentifier": "portrait",
    "elementValue": h'ffd8ffe000104a464946000101010090009000000ffdb004300130d0e110e0c
    13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241435f82
    606871757b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f767676
    76767676767676767676767676767676767676767676767676767676767676767676767676767676
    76767676767676ffc000110800180064030122000211010311 01ffc4001b0000030100030100000000
    0000000000000000050604010203 07ffc400321000010303030205020309000000000000010203040
    0005110612211331141551617122410781a1163542527391b2c1f1ffc400150101010000000000000
    0000000000000000001ffc4001a1101010100030100000000000000000000000014111213161ffda
    000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e7709723a1
    d0dae146ddfbb3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb95ff00
    eb9ff14d215de66af089ce44b7dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65d6b95a
    147381ea0d495b933275fe6bba75c114104a8ba410413e983dff004f5af5d34b4b4cde632d0bf1fd
```

```
      1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e856001ebc03c7ce29dd9eef1ef10fc4
      47dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e54528750946ec67048
      50cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493f602a0
      75559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7290875
      db4be88ab384bbbbbfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30ab420a
      7a6d3240c718cf031fa9ef4c9ad550205aa02951df4a1d6c8421b015b769db8c9229837ea2be8b1b
      0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2edbd4584e7af9c64e3f96b9beb28d4ac4093
      1e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4a179c1c1e460938f9149ef
      655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dce15eed9b
      71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c5256a79e
      706d52d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344e847dc
      ac9dcb5abba8d145061d43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a15f71dd
      27d451453e09d1a21450961cbb3ea8a956433b781f1ce33dfed54f0e2b50a2b71d84ed6db18028a2
      8175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a66b726e33de9ea6f1b52b181c760724e
      47b514520a5a28a283ffd9'
      }
    >>),
  24(<<
      {
      "digestID": 10,
      "random": h'577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f',
      "elementIdentifier": "driving_privileges",
      "elementValue": [
        {
          "vehicle_category_code": "A",
          "issue_date": 1004("2020-09-17"),
          "expiry_date": 1004("2031-06-10")
          }
        ]
      }
    >>),
  24(<<
      {
      "digestID": 11,
      "random": h'fa21d3d890af5f4ea2760d08fd9a6256004cd5aa9d5e697ba5873fb0cddd555e',
      "elementIdentifier": "un_distinguishing_sign",
      "elementValue": "I"
      }
    >>)
  ],
"org.iso.18013.5.1.IT": [
  24(<<
      {
      "digestID": 12,
      "random": h'0c3fe75be952ec3c2257031a71f2f54aeabfe7445705cec147fbb2c0f69ad561',
      "elementIdentifier": "sub",
      "elementValue": "3B4hK2m7fA9TdVzqLrGp6W8XyJ1sNtQc"
```

```
        }
      >>),
    24(<<
        {
        "digestID": 13,
        "random": h'd22c6db3dd27e066deb2ace6161e47fc6abc7a87c84a10320f14bc66d6e08d49',
        "elementIdentifier": "verification",
        "elementValue": {
          "trust_framework": "it_wallet",
          "assurance_level": "https://trust-anchor.example.it/loa/high",
          }
        }
      >>)
    ]
  },
"issuerAuth": [
    << {1: -7} >>,
    {
        33: h'30820208308201afa00302010202142eb39c647c81836bcf79fa9cd0b201ec0bf52307300a0
        6082a8648ce3d0403023064310b3009060355040613025553311330110603550408 0c0a43616c6966
        6f726e6961311630140603550407 0c0d53616e204672616e636973636f31133011060355040a0c0a4
        d7920436f6d70616e7931133011060355040 30c0a6d79736974652e636f6d301e170d323530333237
        3135353532305a170d3235303430363135353532305a3064310b3009060355040613025553311330 1
        106035504080c0a43616c69666f726e6961311630140603550407 0c0d53616e204672616e63697363
        6f31133011060355040a0c0a4d7920436f6d70616e7931133011060355040 30c0a6d79736974652e6
        36f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004f33da72d0dd0009b62221b
        0e839099b12dab5e01021124ebf9060422e648f3c3ec6614a86da1e91e552b2ae35e04d3058ae82b5
        c65a7f1f26800cb4499652a09a33f303d303b0603551d1104343032863068747470733a2f2f637265
        64656e7469616c2d6973737565722e6f6964632d666564657261746 96f6e2e6f6e6c696e65300a060
        82a8648ce3d0403020347003044022 04d1f0819971652b79ebe4825547de3d5554d2f41410225e6b1
        3dab949cda125e022079ba71b823619e49719dce5daa565bf745d3d97e2b87c7f7d6a626f981e653ed'
    },
    << 24(<<
        {
            "version": "1.0",
            "digestAlgorithm": "SHA-256",
            "valueDigests": {
              "org.iso.18013.5.1": {
                  0: h'f46b65d5060ad060ab9be62ff22ea8633437619ebdc7fa81f2d151159e92bffe',
                  1: h'e506545f6a6fd5d982670b4d62fc2b0688dc8f26754e7b0c574d63f5d72a85ac',
                  2: h'cfcf96fa12d100eeed5f00183d3b6a0888baa47eae85b5b95037eca7bbc0d07e',
                  3: h'8b0772252b0e06b611676b6b3402eb33bf866eb145e49f4d5f23215e6a047772',
                  4: h'14135c96693e2ab08d956876ee491357d906a6dd125557196dfb9811ba54aa8d',
                  5: h'86dcbd99233fbb84a9a2dce3a864a425e6e809300067a4475e3ea2a4d233dc74',
                  6: h'2e9512d35ea225e69e7b2180ecc1678dcc3e77a16e36427e64b4f0e2861b4d3a',
                  7: h'4efe55c36f6249d23c473a125afc5181aa3063393649478155 4971b72ff13700',
                  8: h'cc44a4f9983c5b0b1efc0e82e2867c8d5bbdf89c34bff16a1953c923bb4e4b3e',
```

```
                9: h'775eb2af0aa55f2071d62662b35c99698ae3bc0e2c4af5724ff88476cddd152f',
               10: h'915d0ad53dd23dace34968c263d307c04701a9bb9dc9865af91dc409786fd833',
               11: h'47d89ff4fb513044e6f2394236755ac0abf3e4f4a46f40454a458a59f8b7a6fb'
              },
            "org.iso.18013.5.1.IT": {
               12: h'16d2098702e896b4614dff1859bd3b42105cac2e62ce7f87dcacc249a656db32',
               13: h'755fd7c0f9272a8589c4a661a8aa80dc916018e500884eba316899d653fcb8d1'
              }
            },
          "deviceKeyInfo": {
            "deviceKey": {
               1: 2,
              -1: 1,
              -2: h'f96b29873b61f05403e2963a7ecbc799c9aab28d8a6629e5848cfdef85442866',
              -3: h'a9fef033a900c63e3894d8deb805a2a1fb55ef0d2b88e3c0d3336408186485ef'
               }
            },
          "docType": "org.iso.18013.5.1.mDL",
          "validityInfo": {
              "signed": 0("2025-03-27T00:00:00Z"),
              "validFrom": 0("2025-03-27T00:00:00Z"),
              "validUntil": 0("2026-03-27T00:00:00Z")
           },
         "status": {
          "status_list": {
            "idx": 1340,
            "uri": "https://statusprovider.example.org/statuslists/1"
          }
       }
       }
   >>) >>,
   h'd09f9acdf7a6be5e4aeb405bfb3b297b1b8003bcf52558a2f39fc6e5cffed40f18f49d2cc0e72a2a5645
     8d8aade591dee8d6540e639bca637f94bd9fa56f345c'
]
}
```

Acronimi CBOR

Acronimo	Significato
tstr	Text String (Stringa di Testo)
bstr	Byte String (Stringa di Byte)
int	Signed Integer (Intero con Segno)
uint	Unsigned Integer (Intero Senza Segno)
uuid	Universally Unique Identifier (Identificativo Univoco Universale)

continues on next page

Tabella 11.10 – continua dalla pagina precedente

Acronimo	Significato
bool	Boolean (Booleano) (vero/falso)
tdate	Tagged Date (ad esempio, il Tag *0* è usato per indicare una stringa di data/ora in formato **RFC 3339**)

11.1.4 Mappatura dei Parametri degli Attestati Elettronici tra i vari Formati

La seguente tabella fornisce una mappatura comparativa tra le strutture dati degli Attestati Elettronici nei formati SD-JWT VC e mdoc-CBOR. Essa riporta gli elementi e i parametri chiave utilizzati in ciascun formato, evidenziando sia le somiglianze che le differenze. In particolare, evidenzia come i concetti fondamentali - come le informazioni sul Fornitore di Attestati Elettronici, la validità, l'Associazione Crittografica e le disclosure - sono rappresentati nei vari formati possibili di un Attestato Elettronico.

Per SD-JWT-VC, i parametri sono contrassegnati con *(hdr)* se si trovano nell'header JOSE e con *(pld)* se appaiono nel payload del JWT. In mdoc-CBOR, questi parametri sono identificati all'interno delle strutture issuerAuth o nameSpaces.

Informazioni Relative A	Parametri SD-JWT-VC	Parametri mdoc-CBOR
Definizione della Tipologia di Attestato Elettronico Emittente	vct (pld)	issuerAuth.doctype
	iss (pld)	-
	issuing_authority (pld)	nameSpaces.elementIdentifier.issuing_authority
	issuing_country (pld)	nameSpaces.elementIdentifier.issuing_country
Soggetto	sub (pld)	nameSpaces.elementIdentifier.sub
Periodo di validità	iat (pld)	issuerAuth.validityInfo.signed
	exp (pld)	issuerAuth.validityInfo.validUntil
	nbf (pld)	issuerAuth.validityInfo.validFrom
	expiry_date (pld)	nameSpaces.elementIdentifier.expiry_date

continues on next page

Tabella 11.11 – continua dalla pagina precedente

Informazioni Relative A	Parametri SD-JWT-VC	Parametri mdoc-CBOR
Meccanismo di verifica dello stato	status_list (pld)	issuerAuth.status_list
Firma		
	alg (hdr)	issuerAuth.1 (alg)
	kid (hdr)	issuerAuth.4 (kid)
Trust Anchors		
	trust_chain (OID-FED) (hdr)	-
	x5c (hdr)	issuerAuth.33 (x5chain)
Associazione Crittografica	cnf.jwk (pld)	issuerAuth.deviceKeyInfo.deviceKey
Divulgazione Selettiva	_sd_alg (pld)	issuerAuth.digestAlgorithm
	_sd (pld)	issuerAuth.valueDigests
Integrità		
	vct#integrity (pld)	
	Type_Metadata.extends#integrity (hdr)	-
Formato dell'Attestato Elettronico	typ (hdr)	-
Verificabilità dell'Attestato Elettronico	verification (pld)	nameSpaces.elementIdentifier.verification
Disclosure		
	salt	
	claim name	nameSpaces
	claim value	

11.2 Modello di Dati del PID

Il PID è rilasciato dal Fornitore di Attestati Elettronici di Dati di Identificazione Personale secondo le leggi nazionali e DEVE essere fornito in formato SD-JWT VC e mdoc-CBOR.

Nota

Il PID non sarà rilasciato finché l'IT-Wallet non sarà notificato EUDIW; l'**IT-Wallet ID** ha lo scopo di consentire l'accesso ai servizi online dei RP nazionali al posto del PID.

Lo scopo principale del PID è consentire alle persone fisiche di essere autenticate per accedere a un servizio o a una risorsa protetta. Il PID DEVE essere fornito secondo i requisiti del modello dati definiti in EU_2024/2977 e **Sezione 2 dell'ARF PID Rulebook v1.3** [EIDAS-ARF], gli attributi dell'Utente forniti all'interno del PID italiano sono quelli elencati di seguito:

- Cognome attuale
- Nome attuale
- Data di Nascita
- Luogo di Nascita
- Nazionalità
- Ritratto
- Numero amministrativo personale

In aggiunta agli attributi dell'Utente elencati sopra, il PID include anche le seguenti informazioni (EU_2024/2977 e **Sezione 2 dell'ARF PID Rulebook v1.3** [EIDAS-ARF]):

- Autorità emittente
- Paese emittente
- Data di scadenza
- Informazioni sullo stato di validità
- Informazioni di verifica dell'identità e dei dati

Alcuni attributi di dati, come le *informazioni di verifica dell'identità e dei dati*, sono forniti come **estensioni domestiche** definite dalla specifica IT-Wallet italiana. Questi non fanno parte dell'ARF PID Rulebook (Annex 3.01, PID Rulebook v1.3), ma **consentiti, vedi ARF HLR PID_06**, il che consente agli Stati Membri di definire attributi domestici aggiuntivi oltre a quelli specificati nel Regolamento di Esecuzione della Commissione (CIR) 2024/2977 (EU_2024/2977). In particolare, le informazioni di verifica dell'identità sono OBBLIGATORIE per i PID italiani per garantire:

- Tracciabilità del metodo di autenticazione dell'Utente.
- Conformità al livello di garanzia dell'identity proofing durante il processo di enrollment (LoA come definito dal Regolamento eIDAS).
- Verificabilità dei processi di verifica dell'identità e degli attributi dell'Utente.

Gli attributi che sono **estensioni domestiche** DEVONO essere inclusi nei **namespace domestici** che sono definiti nella Sezione *Modello Dati PID in formato SD-JWT-VC* (pagina 405) e Sezione *Modello Dati PID in formato mdoc-CBOR* (pagina 411) per i PID SD-JWT VC e mdoc-CBOR rispettivamente.

11.2.1 Modello Dati PID in formato SD-JWT-VC

Per il PID SD-JWT VC definito in questa specifica, il valore `vct` DEVE essere `urn:eudi:pid:it:1` in conformità ai requisiti dell'ARF PID Rulebook v1.3 per le estensioni domestiche PID (requisito **PID_14**, Sezione 4.2, estendendo il tipo base `urn:eudi:pid:`).

Nota

Fase Transitoria:

Durante la fase transitoria prima della piena operatività EUDIW, le implementazioni nazionali POSSONO utilizzare il valore `vct urn:it-wallet:pid:1`. Una volta raggiunta la piena interoperabilità EUDIW, tutte le implementazioni DEVONO transitare all'identificatore conforme EUDI `urn:eudi:pid:it:1` specificato sopra.

In base a EU_2024/2977 e alla **Sezione 4 dell'ARF PID Rulebook v1.3** [EIDAS-ARF], il PID in formato SD-JWT VC include i seguenti Attributi Utente:

Claim	Descrizione	Riferimento
given_name	OBBLIGATORIO. *Stringa.* Nome attuale.	Sezione 5.1 di OIDC e Regolamento di esecuzione della Commissione EU_2024/2977
family_name	OBBLIGATORIO. *Stringa.* Cognome attuale.	Sezione 5.1 di OIDC e Regolamento di esecuzione della Commissione EU_2024/2977

continues on next page

Tabella 11.12 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
birthdate	OBBLIGATORIO. *Stringa*. Data di Nascita. DEVE essere impostata secondo ISO8601-1 (formato YYYY-MM-DD).	Regolamento di esecuzione della Commissione EU_2024/2977
place_of_birth	OBBLIGATORIO. *Oggetto JSON*. Luogo di Nascita. Almeno uno tra *country*, *region*, *locality* DEVE essere presente.	Regolamento di esecuzione della Commissione EU_2024/2977
nationalities	OBBLIGATORIO. *Array di stringhe*. Uno o più codici paese alpha-2 come specificato in ISO 3166-1.	Regolamento di esecuzione della Commissione EU_2024/2977
picture	OBBLIGATORIO. *Stringa*. Immagine facciale codificata come data URL contenente il ritratto JPEG in base64, conforme ai requisiti di qualità per il tipo di immagine frontale completa definiti in ISO/IEC 39794-5 o, per retrocompatibilità, ISO/IEC 19794-5, clausole 8.2, 8.3 e 8.4. NON DEVE includere le intestazioni o i blocchi specificati nella clausola 5 di ISO/IEC 19794-5, ad eccezione dei soli dati dell'immagine. Salvo il caso in cui l'Utente eserciti esplicitamente l'opt-out, ove applicabile; in caso di opt-out, il valore DEVE essere vuoto, come specificato nell'ARF HLR **PID_03**. L'inclusione obbligatoria dell'attributo portrait si applica a decorrere da 24 mesi dall'entrata in vigore del Regolamento che modifica EU_2024/2977.	Regolamento di esecuzione della Commissione EU_2024/2977 e ARF PID Rulebook
personal_administrative_number	OPZIONALE. *Stringa*. Identificativo univoco nazionale di una persona fisica. DEVE essere conforme con ETSI EN 319 412-1 Sezione 5.1.3. Gli unici identificativi semantici per persone fisiche (**NAT-5.1.3-03**) supportati all'interno di IT-Wallet sono `PNO` e `TIN`. Per IDANPR, DEVE essere usato `PNO`, mentre per il Codice Fiscale, DEVE essere usato `TIN`.	Regolamento di esecuzione della Commissione EU_2024/2977

Nota

Identity Matching and Identity Reconciliation

Per il PID, `personal_administrative_number` è OPZIONALE. Le Relying Party NON DEVONO assumere che un identificativo nazionale univoco sia sempre disponibile. DEVONO applicare le regole di identity matching e identity reconciliation definite in *Identity Matching e Identity Reconciliation* (pagina 567). Quando tale identificativo è presentato, si applica l'Unique National Identifier Binding (*CI_201* (pagina 825)). Quando non è presentato, si applica l'Attribute-Based Binding su almeno `family_name`, `given_name`, `birth_date` (o

`birthdate`) e `place_of_birth` (*CI_202* (pagina 825)).

Tutti gli attributi Utente elencati sopra DEVONO essere divulgabili selettivamente. Oltre agli attributi di metadati obbligatori definiti nella *Tabella Parametri di header JOSE SD-JWT* (pagina 381) e nella *Tabella Parametri SD-JWT* (pagina 381), i seguenti attributi di metadati sono OBBLIGATORI per un PID:

- **date_of_expiry**
- **sub** (estensione domestica)
- **iat**
- **cnf**
- **status**
- **verification** (estensione domestica)

Esempi Non Normativi di PID

Di seguito, l'esempio non normativo del payload di un PID rappresentato in formato JSON.


```
{
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "exp": 1883000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "given_name": "Mario",
  "family_name": "Rossi",
  "birthdate": "1980-01-10",
  "place_of_birth": {
```

```
    "locality": "Roma"
  },
  "nationalities": [
    "IT"
  ],
  "picture": "..."
}
```


La versione SD-JWT corrispondente per il PID è data da


```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
  "_sd": [
    "GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY",
    "Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI",
    "MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk",
    "tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU",
    "uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8",
    "R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A"
  ],
  "exp": 1883000000,
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "_sd_alg": "sha-256",
```

```
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```


L'elenco delle disclosure è presentato di seguito.

Claim `given_name`:

- Hash SHA-256: `Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contenuto: `["kghte5MDNHbQfdJHp88pCA", "given_name", "Mario"]`

Claim `family_name`:

- Hash SHA-256: `MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contenuto: `["hX1TEz_z877_XAtr3COaWg", "family_name", "Rossi"]`

Claim `birthdate`:

- Hash SHA-256: `uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0ZSIsICIxOTgw LTAxLTEwIl0`
- Contenuto: `["YWtI06xDdCyvTalcInTE3A", "birthdate", "1980-01-10"]`

Claim `place_of_birth`:

- Hash SHA-256: `tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU`
- Disclosure: `WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBsYWNlX29mX2JpcnRoIiwg eyJsb2Nhb Gl0eSI6ICJSb21hIn1d`
- Contenuto: `["XcXlPVCqjHNveBCnlVPYgA", "place_of_birth", {"locality": "Roma"}]`

Claim `nationalities`:

- Hash SHA-256: `GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY`
- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBb IklUIl1d`
- Contenuto: `["KNc5-Gk9CQh_TdGbqBKI7A", "nationalities", ["IT"]]`

Claim `picture`:

- Hash SHA-256: `R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A`

- Disclosure: `WyJRaDhMbU40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFn ZS9qc GVnO2Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ`
- Contenuto: `["Qh8LmN4xR7vP2cKjT5sZYA", "picture", "data: image/jpeg; base64,/9j/4AAQSkZJRgABAQAAAQABAAD..."]`

Il formato combinato per l'emissione del PID è dato da:

```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2
N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMi
OiBbIjxJc3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19eyJfc2QiOlsiR0hZanVH
VXRoanRCNHE0T3pfWlNHUG1Db2tMT3B2MmtwRk56ejFMZkZVWSIsIkprYmo4YUxy
LXoyX2MtSFZ4Q2JpdzZZWEZOSGl5TFN2MXhHak44bFJvZ0kiLCJNV0p1ZlF6X0RG
V2M5Y1I0eXhxOFhxbVRaZmdsa2cyRDJTeGEzVUZONFFrIiwidEk1czJBX0V6Nm9a
djZwbFp6VVBqWUFMLVNKR2lBVUZ5UmJoekxzbHVHVSIsInVJYXBVbERUS3NCNXdO
N0JGNnh1Qk5UdGw3NGdsNWlDdV9hUTVuajNZTDgiLCJSNng5bzBqNG0zTDRQcTZ4
WWRLM3JQNG5OcTh2SjJiN3NGcVQ5Y1V3STdBIl0sImV4cCI6MTg4MzAwMDAwMCwi
aXNzIjoiaHR0cHM6Ly9waWRwcm92aWRlci5leGFtcGxlLm9yZyIsInN1YiI6Ik56
YkxzWGg4dURDY2Q3bm9XWEZaQWZIa3hac1JHQzlYcyIsImlhdCI6MTY4MzAwMDAw
MCwiaXNzdWluZ19hdXRob3JpdHkiOiJQSUQgUHJvdmlkZXIgT3JnYW5pemF0aW9u
IiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V4cGlyeSI6IjIwMzMt
MDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4IjoxMjM0LCJ1cmki
OiJodHRwczovL3BpZHByb3ZpZGVyLmV4YW1wbGUub3JnL3N0YXR1cyJ9fSwidmN0
IjoidXJuOmV1ZGk6cGlkOml0OjEiLCJ2Y3QjaW50ZWdyaXR5IjoiNTVmNWZlNTdm
MWU0OTFkNDlmMzM2NzI3ODQ1Mjg1MzJkOWQiLCJ2ZXJpZmljYXRpb24iOnsidHJ1
c3RfZnJhbWV3b3JrIjoiaXRfY2llIiwiYXNzdXJhbmNlX2xldmVsIjoiaHR0cHM6
Ly90cnVzdC1hbmNob3IuZXhhbXBsZS5pdC9sb2EvaGlnaCJ9LCJfc2RfYWxnIjoi
c2hhLTI1NiIsImNuZiI6eyJqd2siOnsia3R5IjoiRUMiLCJjcnYiOiJQLTI1NiIs
IngiOiJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbERsczd2Q2VHZW1j
IiwieSI6Ilp4amlXV2JaTVFHSFZXS1ZRNGhiU0lpcnNWZnVlY0NFNnQ0alQ5RjJI
WlEifX19~WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCA
iTWFyaW8iXQ~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1l
IiwgIlJvc3NpIl0~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0
ZSIsICIxOTgwLTAxLTEwIl0~WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBs
YWNlX29mX2JpcnRoIiwgeyJsb2NhbGl0eSI6ICJSb21hIn1d~WyJLTmM1LUdrOUN
RaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIklUIl1d~WyJRaDhMbU
40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFnZS9qcGVnO2
Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ~
```

11.2.2 Modello Dati PID in formato mdoc-CBOR

Il PID in formato mdoc-CBOR DEVE utilizzare il **docType** `eu.europa.ec.eudi.pid.1` in conformità al requisito ARF **PID_04**.

Gli attributi PID DEVONO essere codificati come specificato nella **Sezione 3 dell'ARF PID Rulebook v1.3** [EIDAS-ARF] e organizzati nei seguenti namespace:

- **Attributi PID standard ARF**: namespace `eu.europa.ec.eudi.pid.1`
- **Estensioni domestiche italiane**: namespace `eu.europa.ec.eudi.pid.it.1`

In base a EU_2024/2977 e alla **Sezione 3 dell'ARF PID Rulebook v1.3** [EIDAS-ARF], il PID in formato mdoc-CBOR include i seguenti Attributi Utente:

elementIdentifier	Descrizione	Namespace
given_name	OBBLIGATORIO. *(tstr)*. Nome attuale.	`eu.europa.ec.eudi.pid.1`
family_name	OBBLIGATORIO. *(tstr)*. Cognome attuale.	`eu.europa.ec.eudi.pid.1`
birth_date	OBBLIGATORIO. *(full-date)*. Data di Nascita. DEVE essere codificata come stringa full-date secondo **RFC 8949**.	`eu.europa.ec.eudi.pid.1`
place_of_birth	OBBLIGATORIO. *(map)*. Luogo di Nascita. Almeno uno tra `country`, `region`, `locality` DEVE essere presente.	`eu.europa.ec.eudi.pid.1`
nationality	OBBLIGATORIO. *(array di tstr)*. Uno o più codici paese alpha-2 come specificato in ISO 3166-1. Codificato come tipo CDDL `nationalities` (array di codici paese).	`eu.europa.ec.eudi.pid.1`
portrait	OBBLIGATORIO. *(bstr)*. Immagine facciale in formato JPEG, conforme ai requisiti di qualità per il tipo di immagine frontale completa definiti in ISO/IEC 39794-5 o, per retrocompatibilità, ISO/IEC 19794-5, clausole 8.2, 8.3 e 8.4. NON DEVE includere le intestazioni o i blocchi specificati nella clausola 5 di ISO/IEC 19794-5, ad eccezione dei soli dati dell'immagine. Salvo il caso in cui l'Utente eserciti esplicitamente l'opt-out, ove applicabile; in caso di opt-out, il valore DEVE essere vuoto, come specificato nell'ARF HLR **PID_03**. L'inclusione obbligatoria dell'attributo portrait si applica a decorrere da 24 mesi dall'entrata in vigore del Regolamento che modifica EU_2024/2977.	`eu.europa.ec.eudi.pid.1`
personal_administrative_number	OPZIONALE. *(tstr)*. Identificativo univoco nazionale di una persona fisica generato da ANPR.	`eu.europa.ec.eudi.pid.1`

Oltre agli attributi di metadati obbligatori definiti nella *Tabella MobileSecurityObject* (pagina 391) e nella *Tabella Attributi Metadata mdoc-CBOR* (pagina 393), i seguenti attributi di metadati sono OBBLIGATORI per un PID:

Attributo	Posizione
expiry_date	namespace `eu.europa.ec.eudi.pid.1`
sub	namespace `eu.europa.ec.eudi.pid.it.1`
validityInfo.signed	MobileSecurityObject
verification	namespace `eu.europa.ec.eudi.pid.it.1`
status	MobileSecurityObject (come definito nella Sezione 6.3 di TOKEN-STATUS-LIST)

Nota

Differenze chiave rispetto alla codifica SD-JWT:

L'ARF PID Rulebook v1.3 utilizza nomi di claim diversi tra i formati SD-JWT e mdoc-CBOR:

- mdoc usa `birth_date` (non `birthdate` come in SD-JWT)
- mdoc usa `expiry_date` (non `date_of_expiry` come in SD-JWT)
- mdoc usa `nationality` (non `nationalities` come in SD-JWT). Nota: entrambi i formati codificano il valore come array di codici paese.
- mdoc usa `portrait` (non `picture` come in SD-JWT). Nota: SD-JWT codifica il valore come data URL; mdoc codifica i byte JPEG come `bstr`.

Vedere la Sezione 3.1.1 (codifica mdoc) e la Sezione 4.1.1 (codifica SD-JWT) dell'ARF PID Rulebook v1.3 per la mappatura completa.

Esempio non normativo del PID in mdoc-CBOR

Un esempio non normativo di un PID in formato mdoc-CBOR (notazione diagnostica) è mostrato di seguito:

```
{
   "nameSpaces": {
       "eu.europa.ec.eudi.pid.1": [
       24(<<
           {
           "digestID": 1,
           "random": h'E0B70BCEFBD43686F345C9ED429343AA',
           "elementIdentifier": "expiry_date",
           "elementValue": 1004("2024-02-22")
           }
       >>),
       24(<<
```

```
    {
    "digestID": 2,
    "random": h'960CB15A2EA9B68E5233CE902807AA95',
    "elementIdentifier": "issuing_country",
    "elementValue": "IT"
    }
>>),
24(<<
    {
    "digestID": 3,
    "random": h'9D3774BD5994CCFED248674B32A4F76A',
    "elementIdentifier": "issuing_authority",
    "elementValue": "Ministero dell'Interno"
    }
>>),
24(<<
    {
    "digestID": 4,
    "random": h'EB12193DC66C6174530CDC29B274381F',
    "elementIdentifier": "given_name",
    "elementValue": "Mario"
    }
>>),
24(<<
    {
    "digestID": 5,
    "random": h'DB143143538F3C8D41DC024F9CB25C9D',
    "elementIdentifier": "family_name",
    "elementValue": "Rossi"
    }
>>),
24(<<
    {
    "digestID": 6,
    "random": h'6059FF1CE27B4997B4ADE1DE7B01DC60',
    "elementIdentifier": "birth_date",
    "elementValue": 1004("1956-01-12")% the tag 1004 defines the value
    is a full date
    }
>>),
24(<<
    {
    "digestID": 7,
    "random": h'FX8VVN1YN6NBH3CKYPJTNPJIVJ5KDMFA',
    "elementIdentifier": "place_of_birth",
    "elementValue": {"locality": "Roma"}
    }
```

```
        >>),
        24(<<
            {
            "digestID": 8,
            "random": h'RGLXZ5CHND8Y2UUMHXXMJWPCHNH9HVIX',
            "elementIdentifier": "nationality",
            "elementValue": ["IT"],
            }
        >>),
        24(<<
            {
            "digestID": 9,
            "random": h'71EAC48D2F9B635A0C17D4E82ABF05',
            "elementIdentifier": "portrait",
            "elementValue": h'FFD8FFE000104A4649460001010000010001000 0...'
            }
        >>)
        ],
        "eu.europa.ec.eudi.pid.it.1": [
            24(<<
                {
                "digestID": 10,
                "random": h'44DD0606C5F5FDD206H8C84H3F67F5HEEH',
                "elementIdentifier": "sub",
                "elementValue": "550e8400-e29b-41d4-a716-446655440000"
                }
            >>),
            24(<<
                {
                "digestID": 11,
                "random": h'33cc9495c4f4fcc195g7b73g2e56e4gddg',
                "elementIdentifier": "verification",
                "elementValue": {
                    "trust_framework": "it_cie",
                    "assurance_level": "https://trust-anchor.example.it/loa/high",
                }
                }
            >>)
        ]
    },
    "issuerAuth": [
          << {1: -7} >>, % protected header with the value alg:ES256
          {
              33: h'30820215308201BCA003020102021404AD30C…'% 33->X5chain:COSE X_509
          },
          <<
              24(<<
```

```
                {
                "docType": "eu.europa.ec.eudi.pid.1",
                "version": "1.0",
                "validityInfo": {
                    "signed": 0("2023-02-22T06:23:56Z"),
                    "validFrom": 0("2023-02-22T06:23:56Z"),
                    "validUntil": 0("2024-02-22T00:00:00Z")
                },
                "valueDigests": {
                    "eu.europa.ec.eudi.pid.1": {
                        1: h'0F1571A97FFB799CC8FCDF2BA4FC2909929…',
                        2: h'E2382149255AE8E955AF9B8984395…',
                        3: h'BBC77E6CCA981A3AD0C3E544EDF86…',
                        4: h'AA5D5C57B6E7D88FC6F98895E4F67…',
                        5: h'BB6E6C68D1B4B4EC5A2AE9206F5t4…',
                        6: h'F8A5966E6DAC9970E0334D8F75E25…',
                        7: h'1F1A0F7CCDD79b29EB4B22F96F54F…',
                        8: h'1A0F7EEDB46B37AS4N22R96D83F3F…',
                        9: h'71EAC48D2F9B635A0C17D4E82ABF5…'
                    },
                    "eu.europa.ec.eudi.pid.it.1": {
                        10: h'C8FF5E58G89FCE97G45533CE3E4B…',
                        11: h'F9EE4D36F67DBD75E23311AC1C29…',
                        12: h'A2BC5E47G78ECD86F34422BD2D3A…'
                    }
                },
                "deviceKeyInfo": {
                    "deviceKey": {
                        1: 2, % kty:EC2 (Eliptic curves with x and y coordinate pairs)
                        -1: 1, % crv:p256
                        -2: h'B820963964E53AF064686DD9218303494A…', % x-coordiantes
                        -3: h'0A6DA0AF437E2943F1836F31C678D89298E9…'% y-ccordiantes
                    }
                },
                "status": {
                    "status_list": {
                        "idx": 0,
                        "uri": "https://issuer.example.org/statuslists/1"
                    }
                },
                "digestAlgorithm": "SHA-256"
                }
            >>)
        >>,
        h'1AD0D6A7313EFDC38FCD765852FA2BD43DEBF48BF5A580D'
        ]
}
```

11.3 Modello di Dati dell'IT-Wallet ID

L'**IT-Wallet ID** (*Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale*) è un Attestato Elettronico di Attributi (EAA) rilasciato dall'EAA Provider secondo le leggi nazionali. È fornito in formato SD-JWT VC.

L'IT-Wallet ID è destinato **esclusivamente all'uso nazionale** con Relying Party che operano nell'ambito della giurisdizione nazionale per l'accesso ai servizi online. NON DEVE essere utilizzato per interazioni cross-border e **non** costituisce un PID ai sensi del quadro europeo di Identità Digitale.

Lo scopo principale dell'IT-Wallet ID è consentire alle persone fisiche di essere autenticate per l'accesso a un servizio o a una risorsa protetta nell'ambito della giurisdizione nazionale. Gli attributi dell'Utente forniti all'interno dell'IT-Wallet ID sono quelli elencati di seguito:

- Cognome attuale
- Nome attuale
- Data di Nascita
- Luogo di Nascita
- Nazionalità
- Numero di identificazione dell'Utente nei servizi delle Relying Party pubbliche (ad esempio il *codice fiscale*)

In aggiunta agli attributi dell'Utente elencati sopra, l'IT-Wallet ID include anche i seguenti attributi di metadati:

- Autorità emittente
- Paese emittente
- Data di scadenza
- Informazioni sullo stato di validità
- Informazioni di verifica dell'identità e dei dati

Le informazioni di identity proofing sono OBBLIGATORIE per l'IT-Wallet ID al fine di garantire:

- La valutazione del metodo di autenticazione dell'Utente utilizzato.
- La conformità al livello di garanzia dell'identity proofing durante il processo di iscrizione (*enrollment*), secondo il LoA definito dal Regolamento eIDAS.
- L'auditabilità dei processi di verifica degli attributi dell'Utente.

11.3.1 Modello di Dati dell'IT-Wallet ID in formato SD-JWT VC

L'IT-Wallet ID SD-JWT VC definito in questa specifica DEVE utilizzare il valore del claim `vct` impostato a `urn:it-wallet:eid:1`, dove `eid` è il `credential_type` di catalogo dell'IT-Wallet ID.

L'IT-Wallet ID in formato SD-JWT VC include i seguenti Attributi Utente:

Claim	Descrizione	Riferimento
given_name	OBBLIGATORIO. *Stringa*. Nome attuale.	Sezione 5. 1 di OIDC e Regolamento di Esecuzione della Commissione EU_ 2024/2977
family_name	OBBLIGATORIO. *Stringa*. Cognome attuale.	Sezione 5. 1 di OIDC e Regolamento di Esecuzione della Commissione EU_ 2024/2977
birthdate	OBBLIGATORIO. *Stringa*. Data di Nascita. DEVE essere impostata secondo ISO8601-1 (formato YYYY-MM-DD).	Regolamento di Esecuzione della Commissione EU_ 2024/2977
place_of_birth	OBBLIGATORIO. *Oggetto JSON*. Luogo di Nascita. Almeno uno tra *country*, *region*, *locality* DEVE essere presente.	Regolamento di Esecuzione della Commissione EU_ 2024/2977
nationalities	OBBLIGATORIO. *Array di stringhe*. Uno o più codici paese alpha-2 come specificato in ISO 3166-1.	Regolamento di Esecuzione della Commissione EU_ 2024/2977
personal_ administrative_ number	OBBLIGATORIO se `tax_id_code` non è presente, OPZIONALE altrimenti. *Stringa*. Identificativo nazionale univoco di una persona fisica generato da ANPR in formato stringa.	Regolamento di Esecuzione della Commissione EU_ 2024/2977
tax_id_code	OBBLIGATORIO se `personal_administrative_number` non è presente, OPZIONALE altrimenti. *Stringa*. Codice fiscale nazionale della persona fisica in formato stringa. DEVE essere impostato secondo ETSI EN 319 412-1. Ad esempio `TINIT-<ItalianTaxIdentificationNumber>`.	Estensione domestica

Nota

Identity Matching

Per l'IT-Wallet ID è presente almeno uno tra `tax_id_code` e `personal_administrative_number`. La Relying Party DEVE applicare l'Unique National Identifier Binding come specificato in *Identity Matching e Identity Reconciliation* (pagina 567) (*CI_201* (pagina 825)).

Tutti gli attributi dell'Utente elencati sopra DEVONO essere divulgabili selettivamente (*selectively disclosable*). In aggiunta agli attributi di metadati obbligatori definiti nella *Tabella Parametri di header JOSE SD-JWT* (pagina 381) e nella *Tabella Parametri SD-JWT* (pagina 381), i seguenti attributi di metadati sono OBBLIGATORI per un IT-Wallet ID:

- **date_of_expiry**
- **sub** (estensione domestica)
- **iat**
- **cnf**
- **status**
- **verification** (estensione domestica)

Esempio Non Normativo di IT-Wallet ID in SD-JWT-VC

Di seguito, l'esempio non normativo del payload di un IT-Wallet ID rappresentato in formato JSON.


```
{
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "exp": 1883000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
```

```
  },
  "given_name": "Mario",
  "family_name": "Rossi",
  "birthdate": "1980-01-10",
  "place_of_birth": {
    "locality": "Roma"
  },
  "nationalities": [
    "IT"
  ],
  "picture": "..."
}
```


La corrispondente versione SD-JWT per l'IT-Wallet ID è data da


```
{
    "alg": "ES256",
    "typ": "dc+sd-jwt",
    "kid": "dB67gL7ck3TFiIAf7N6_7SHvqk0MDYMEQcoGGlkUAAw",
    "x5c": [
        "<Issuer X.509 Certificate>"
    ]
}
```

```
{
  "_sd": [
    "GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY",
    "Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI",
    "MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk",
    "tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU",
    "uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8",
    "R6x9o0j4m3L4Pq6xYdK3rP4nNq8vJ2b7sFqT9cUwI7A"
  ],
  "exp": 1883000000,
  "iss": "https://pidprovider.example.org",
  "sub": "NzbLsXh8uDCcd7noWXFZAfHkxZsRGC9Xs",
  "iat": 1683000000,
  "issuing_authority": "PID Provider Organization",
  "issuing_country": "IT",
  "date_of_expiry": "2033-03-19",
  "status": {
    "status_list": {
      "idx": 1234,
      "uri": "https://pidprovider.example.org/status"
    }
  },
  "vct": "urn:eudi:pid:it:1",
  "vct#integrity": "55f5fe57f1e491d49f33672784528532d9d",
```

```
  "verification": {
    "trust_framework": "it_cie",
    "assurance_level": "https://trust-anchor.example.it/loa/high"
  },
  "_sd_alg": "sha-256",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```


L'elenco delle disclosure è presentato di seguito.

Claim `given_name`:

- SHA-256 Hash: `Jkbj8aLr-z2_c-HVxCbiw6YXFNHiyLSv1xGjN8lRogI`
- Disclosure: `WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCAiTWFy aW8iXQ`
- Contents: `["kghte5MDNHbQfdJHp88pCA", "given_name", "Mario"]`

Claim `family_name`:

- SHA-256 Hash: `MWJufQz_DFWc9cR4yxq8XqmTZfglkg2D2Sxa3UFN4Qk`
- Disclosure: `WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1lIiwgIlJv c3NpIl0`
- Contents: `["hX1TEz_z877_XAtr3COaWg", "family_name", "Rossi"]`

Claim `birthdate`:

- SHA-256 Hash: `uIapUlDTKsB5wN7BF6xuBNTtl74gl5iCu_aQ5nj3YL8`
- Disclosure: `WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0ZSIsICIxOTgw LTAxLTEwIl0`
- Contents: `["YWtI06xDdCyvTalcInTE3A", "birthdate", "1980-01-10"]`

Claim `tax_id_code`:

- SHA-256 Hash: `_C7hoKFt0kV190v2GXIwLUIiDbc_7LcyofQmgDfute8`
- Disclosure: `WyItejM0Y0oxZ0M1VUJQQ0l4OE9oTmlRIiwgInRheF9pZF9jb2RlIiwgIlRJ Tkl ULVhYWFhYWFhYWFhYWFhYWFgiXQ`
- Contents: `["-z34cJ1gC5UBPCIx8OhNiQ", "tax_id_code", "TINIT-XXXXXXXXXXXXXXXX"]`

Claim `place_of_birth`:

- SHA-256 Hash: `tI5s2A_Ez6oZv6plZzUPjYAL-SJGiAUFyRbhzLsluGU`

- Disclosure: `WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBsYWNlX29mX2JpcnRoIiwg eyJsb2NhbGl0eSI6ICJSb21hIn1d`
- Contents: `["XcXlPVCqjHNveBCnlVPYgA", "place_of_birth", {"locality": "Roma"}]`

Claim `nationalities`:

- SHA-256 Hash: `GHYjuGUthjtB4q4Oz_ZSGPmCokLOpv2kpFNzz1LfFUY`
- Disclosure: `WyJLTmM1LUdrOUNRaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBb IklUIl1d`
- Contents: `["KNc5-Gk9CQh_TdGbqBKI7A", "nationalities", ["IT"]]`

Il formato combinato per l'emissione dell'IT-Wallet ID è dato da:


```
eyJhbGciOiAiRVMyNTYiLCAidHlwIjogImRjK3NkLWp3dCIsICJraWQiOiAiZEI2
N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIsICJ4NWMi
OiBbIjxJc3N1ZXIgWC41MDkgQ2VydGlmaWNhdGU-Il19eyJfc2QiOlsiR0hZanVH
VXRoanRCNHE0T3pfWlNHUG1Db2tMT3B2MmtwRk56ejFMZkZVWSIsIkprYmo4YUxy
LXoyX2MtSFZ4Q2JpdzZZWEZOSGl5TFN2MXhHak44bFJvZ0kiLCJNV0p1ZlF6X0RG
V2M5Y1I0eXhxOFhxbVRaZmdsa2cyRDJTeGEzVUZONFFrIiwidEk1czJBX0V6Nm9a
djZwbFp6VVBqWUFMLVNKR2lBVUZ5UmJoekxzbHVHVSIsInVJYXBVbERUS3NCNXdO
N0JGNnh1Qk5UdGw3NGdsNWlDdV9hUTVuajNZTDgiLCJSNng5bzBqNG0zTDRQcTZ4
WWRLM3JQNG5OcTh2SjJiN3NGcVQ5Y1V3STdBIl0sImV4cCI6MTg4MzAwMDAwMCwi
aXNzIjoiaHR0cHM6Ly9waWRwcm92aWRlci5leGFtcGxlLm9yZyIsInN1YiI6Ik56
YkxzWGg4dURDY2Q3bm9XWEZaQWZIa3hac1JHQzlYcyIsImlhdCI6MTY4MzAwMDAw
MCwiaXNzdWluZ19hdXRob3JpdHkiOiJQSUQgUHJvdmlkZXIgT3JnYW5pemF0aW9u
IiwiaXNzdWluZ19jb3VudHJ5IjoiSVQiLCJkYXRlX29mX2V4cGlyeSI6IjIwMzMt
MDMtMTkiLCJzdGF0dXMiOnsic3RhdHVzX2xpc3QiOnsiaWR4IjoxMjM0LCJ1cmki
OiJodHRwczovL3BpZHByb3ZpZGVyLmV4YW1wbGUub3JnL3N0YXR1cyJ9fSwidmN0
IjoidXJuOmV1ZGk6cGlkOml0OjEiLCJ2Y3QjaW50ZWdyaXR5IjoiNTVmNWZlNTdm
MWU0OTFkNDlmMzM2NzI3ODQ1Mjg1MzJkOWQiLCJ2ZXJpZmljYXRpb24iOnsidHJ1
c3RfZnJhbWV3b3JrIjoiaXRfY2llIiwiYXNzdXJhbmNlX2xldmVsIjoiaHR0cHM6
Ly90cnVzdC1hbmNob3IuZXhhbXBsZS5pdC9sb2EvaGlnaCJ9LCJfc2RfYWxnIjoi
c2hhLTI1NiIsImNuZiI6eyJqd2siOnsia3R5IjoiRUMiLCJjcnYiOiJQLTI1NiIs
IngiOiJUQ0FFUjE5WnZ1M09IRjRqNFc0dmZTVm9ISVAxSUxpbERsczd2Q2VHZW1j
IiwieSI6Ilp4amlXV2JaTVFHSFZXS1ZRNGhiU0lpcnNWZnVlY0NFNnQ0alQ5RjJI
WlEifX19~WyJrZ2h0ZTVNRE5IYlFmZEpIcDg4cENBIiwgImdpdmVuX25hbWUiLCA
iTWFyaW8iXQ~WyJoWDFURXpfejg3N19YQXRyM0NPYVdnIiwgImZhbWlseV9uYW1l
IiwgIlJvc3NpIl0~WyJZV3RJMDZ4RGRDeXZUYWxjSW5URTNBIiwgImJpcnRoZGF0
ZSIsICIxOTgwLTAxLTEwIl0~WyJYY1hsUFZDcWpITnZlQkNubFZQWWdBIiwgInBs
YWNlX29mX2JpcnRoIiwgeyJsb2NhbGl0eSI6ICJSb21hIn1d~WyJLTmM1LUdrOUN
RaF9UZEdicUJLSTdBIiwgIm5hdGlvbmFsaXRpZXMiLCBbIklUIl1d~WyJRaDhMbU
40eFI3dlAyY0tqVDVzWllBIiwgInBpY3R1cmUiLCAiZGF0YTppbWFnZS9qcGVnO2
Jhc2U2NCwvOWovNEFBUVNrWkpSZ0FCQVFBQUFRQUJBQUQuLi4iXQ~
```

11.4 Ciclo di Vita degli Attestati Elettronici

Il Fornitore di Attestati Elettronici è responsabile della creazione e del rilascio degli Attestati Elettronici, nonché della gestione del loro ciclo di vita e dello stato di validità.

La Fonte Autentica è l'entità responsabile della gestione e della fornitura degli Attributi dell'Utente ai Fornitori di Attestati Elettronici. Esiste una relazione tra il ciclo di vita degli attributi gestiti dalla Fonte Autentica e il ciclo di vita degli Attestati Elettronici gestito dal Fornitore di Attestati Elettronici. Infatti, uno dei motivi per la revoca o la sospensione degli Attestati Elettronici è l'aggiornamento/revoca o sospensione degli attributi contenuti nell'Attestato Elettronico. In IT Wallet, la fornitura degli Attributi dell'Utente e la notifica di aggiornamenti o modifiche dei relativi stati vengono gestiti utilizzando l'infrastruttura PDND (vedere le relative sezioni per maggiori dettagli).

La Fig. 11.1 mostra gli stati e le transizioni relativi agli Attestati Elettronici. Include quattro stati distinti: **Issued**, **Valid**, **Expired** e **Revoked**. Mentre, nel caso degli Attestati Elettronici di Attributi (Q)EAA, c'è uno stato aggiuntivo: **Suspended**. Indipendentemente dallo stato, un Attestato Elettronico può essere eliminato (**PID/(Q)EAA DEL**) e questo termina il suo ciclo di vita.

Nota

Gli Utenti POSSONO presentare un Attestato Elettronico in qualsiasi stato, spetta alla policy della Relying Party accettare un Attestato Elettronico non Valido. Un esempio di questo scenario è quando una Relying Party deve verificare che l'Utente non sia minorenne. In questo caso, anche se l'Utente presenta un Attestato Elettronico **Issued/Expired/Revoked** o **Suspended**, l'attributo relativo all'età è ancora affidabile.

Nota

Mentre **Issued**, **Valid**, **Expired**, **Revoked** sono esplicitamente menzionati nell'ARF (vedi Figura 5 di ARF v1.4), **Suspended** è implicitamente presente in EIDAS-ARF. Questa specifica lo considera esplicitamente.

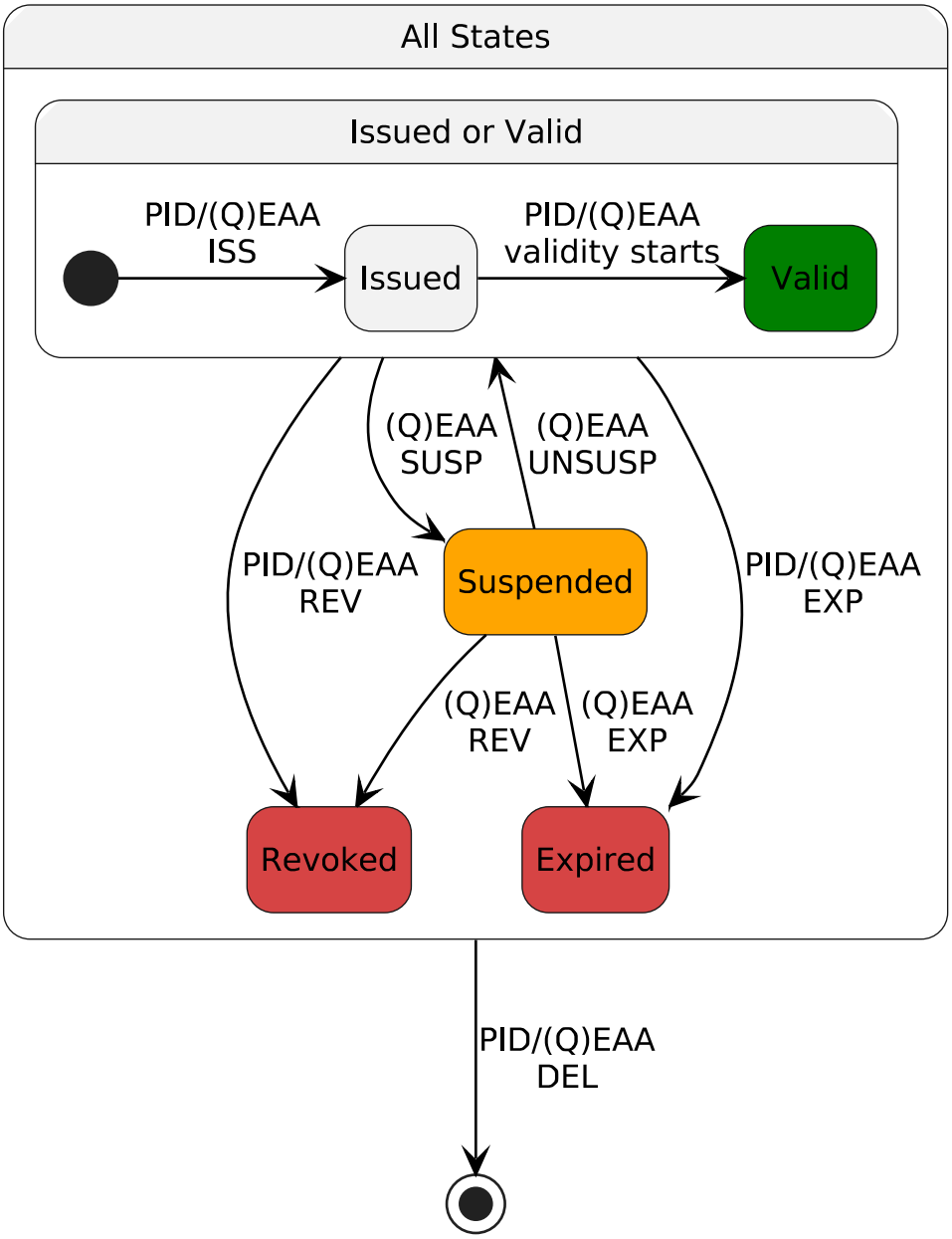


Fig. 11.1: Transazioni di Stato degli Attestati Elettronici.

11.4.1 Transizioni degli Attestati Elettronici

Transizione dell'Attestato Elettronico a Issued

Affinché la macchina a stati possa avviarsi, l'Istanza del Wallet DEVE essere nello stato **Operational** o **Valid**, consentendo il rilascio di Attestati Elettronici. La macchina a stati inizia con lo stato **Issued**, quando viene attivato un processo di emissione e, di conseguenza, un Attestato Elettronico viene rilasciato all' Istanza del Wallet (**PID/(Q)EAA ISS**). Si prega di fare riferimento a *Emissione di Attestati Elettronici* (pagina 443).

Transizione dell'Attestato Elettronico a Valid

Un Attestato Elettronico passa allo stato **Valid** quando:

- raggiunge la sua data di inizio validità;
- viene attivato un processo di riattivazione se l'Attestato Elettronico di Attributi (Q)EAA è stato sospeso.

Transizione dell'Attestato Elettronico a Expired

Un Attestato Elettronico passa naturalmente allo stato **Expired** quando scade automaticamente al raggiungimento della sua data di fine validità (**PID/(Q)EAA EXP**), indicando che non è più valido per l'uso.

Se un Attestato Elettronico è **Expired**, l'Istanza del Wallet DOVREBBE notificare all'Utente che l'Attestato Elettronico è scaduto e l'Utente PUÒ eliminarlo (**PID/(Q)EAA DEL**). Questo termina il suo ciclo di vita.

Transizione dell'Attestato Elettronico a Revoked

Un Attestato Elettronico passa dagli stati **Issued**, **Valid** o **Suspended** allo stato **Revoked** quando viene attivamente revocato dal Fornitore di Attestati Elettronici tramite un processo di revoca (**PID/(Q)EAA REV**). Le Relying Party NON DOVREBBERO più considerare utilizzabile un particolare Attestato Elettronico quando è **Revoked**, anche se è ancora temporalmente valido e contiene una firma valida del Fornitore di Attestati Elettronici. La revoca può avvenire nei seguenti casi:

- per motivi tecnici di sicurezza relativi alla compromissione del materiale crittografico;
- in caso di richieste esplicite dell'Utente;
- come conseguenza di un aggiornamento degli attributi da parte delle Fonti Autentiche;

- in caso di revoca degli attributi contenuti nell'Attestato Elettronico notificata dalla Fonte Autentica;
- morte dell'Utente;
- revoca dell'Istanza del Wallet a cui è stato rilasciato l'Attestato Elettronico;
- attività illegali dell'Utente segnalate da Organi Giudiziari o di Vigilanza.

Nel caso del solo PID e IT-Wallet ID, i seguenti casi si aggiungono a quelli sopra elencati:

- rilevamento di una violazione dell'identità digitale rilasciata da un Gestore di Identità Digitale e utilizzata per autenticare l'Utente durante il rilascio del PID/IT-Wallet ID;
- come risultato dell'ottenimento di un nuovo PID/IT-Wallet ID su una nuova Istanza del Wallet dallo stesso Fornitore di Wallet che ha fornito l'Istanza del Wallet contenente un PID/IT-Wallet ID precedentemente rilasciato.

Nota

Un Fornitore di Attestati Elettronici di Attributi (Q)EAA PUÒ revocare un Attestato Elettronico di Attributi (Q)EAA in caso di revoca del PID/IT-Wallet ID.

Quando un Attestato Elettronico è **Revoked** non può tornare allo stato **Valid**, l'Istanza del Wallet DOVREBBE notificare all'Utente che l'Attestato Elettronico è stato revocato e l'Utente PUÒ eliminarlo (**PID/(Q)EAA DEL**). Questo termina il suo ciclo di vita.

Transizione dell'Attestato Elettronico a Suspended

Un Attestato Elettronico di Attributi (Q)EAA passa dagli stati **Issued** o **Valid** allo stato **Suspended** quando viene sospeso dal Fornitore di Attestati Elettronici (**(Q)EAA SUSP**). L'Attestato Elettronico di Attributi (Q)EAA rimane **Suspended** fino a quando non viene ripristinato allo stato **Issued** o **Valid** (**(Q)EAA UNSUSP**) a seconda dello stato precedente, cioè le condizioni che hanno portato alla sua sospensione vengono risolte, o passa a **Revoked**, **Expired** o viene eliminato. La sospensione di un Attestato Elettronico di Attributi (Q)EAA PUÒ essere:

- Guidata dal caso d'uso, basata sullo stato di validità degli attributi contenuti nell'Attestato Elettronico di Attributi (Q)EAA. In questo caso, una Fonte Autentica aderente a PDND DEVONO notificare al Fornitore di Attestati Elettronici qualsiasi cambiamento nello stato degli attributi attestati dall'Attestato Elettronico di Attributi (Q)EAA.
- Esplicitamente richiesta dall'Utente.

11.4.2 Ciclo di vita degli Attestati Elettronici ottenuti in batch

Ciascuna degli Attestati Elettronici emessi in batch, entra immediatamente nel proprio stato relativo al ciclo di vita. Tutte le transizioni di stato (Emessa → Valida → Scaduta/Sospesa/ Revocata) avvengono a livello di singolo Attestato Elettronico, utilizzando i parametri individuali dell'Attestato Elettronico (ad esempio, date di validità, Status List).

11.4.3 Gestione del Ciclo di Vita degli Attestati Elettronici

Mentre la Fig. 11.1 mostra i diversi stati che un Attestato Elettronico può acquisire durante il suo ciclo di vita, la Fig. 11.2 mostra il punto di vista delle Istanze del Wallet e dei Fornitori di Attestati Elettronici nella gestione del ciclo di vita degli Attestati Elettronici e l'effetto sul loro storage locale.

Un Utente, attraverso l'Istanza del Wallet, è in grado di acquisire un nuovo Attestato Elettronico (**Credential Acquisition**) eseguendo il processo **PID/(Q)EAA ISS**. Questo DEVE risultare nella memorizzazione di un Attestato Elettronico nello stato **Issued/Valid**, ed sua eliminazione quando non è più necessario o è **Expired/Revoked** (**Credential Deletion**). Fino alla **Credential Deletion**, un Attestato Elettronico può essere presentato alle Relying Party, questa operazione non influenzerà il suo ciclo di vita.

Un Fornitore di Attestati Elettronici invece è responsabile per:

- **Generazione dell'Attestato Elettronico**: l'Attestato Elettronico viene generato come conseguenza di una richiesta di emissione e DEVE essere aggiunto allo storage locale del Fornitore di Attestati Elettronici dopo l'emissione riuscita.
- **Revoca/Sospensione/Riattivazione dell'Attestato Elettronico** (**PID/(Q)EAA REV** e **(Q)EAA SUSP/UNSUSP**): per motivi di sicurezza tecnici o richiesti da entità esterne (ad esempio, Utenti e Fonti Autentiche) lo stato dell'Attestato Elettronico DEVE essere aggiornato localmente.
- **Eliminazione dei Dati**: dopo aver raggiunto lo stato **Expired**, e in base alle politiche di conservazione del Fornitore di Attestati Elettronici, gli Attestati Elettronici DEVONO essere rimossi dallo storage locale del Fornitore di Attestati Elettronici.

Revoca e Sospensione degli Attestati Elettronici

Questa sezione descrive i flussi per richiedere un aggiornamento dello stato di un Attestato Elettronico (cioè revoca o sospensione), le entità coinvolte e i meccanismi di validazione per gli Attestati Elettronici nel sistema IT-Wallet.

Come evidenziato nella Sezione *Ciclo di Vita degli Attestati Elettronici* (pagina 422), il ciclo di vita di un Attestato Elettronico è influenzato da:

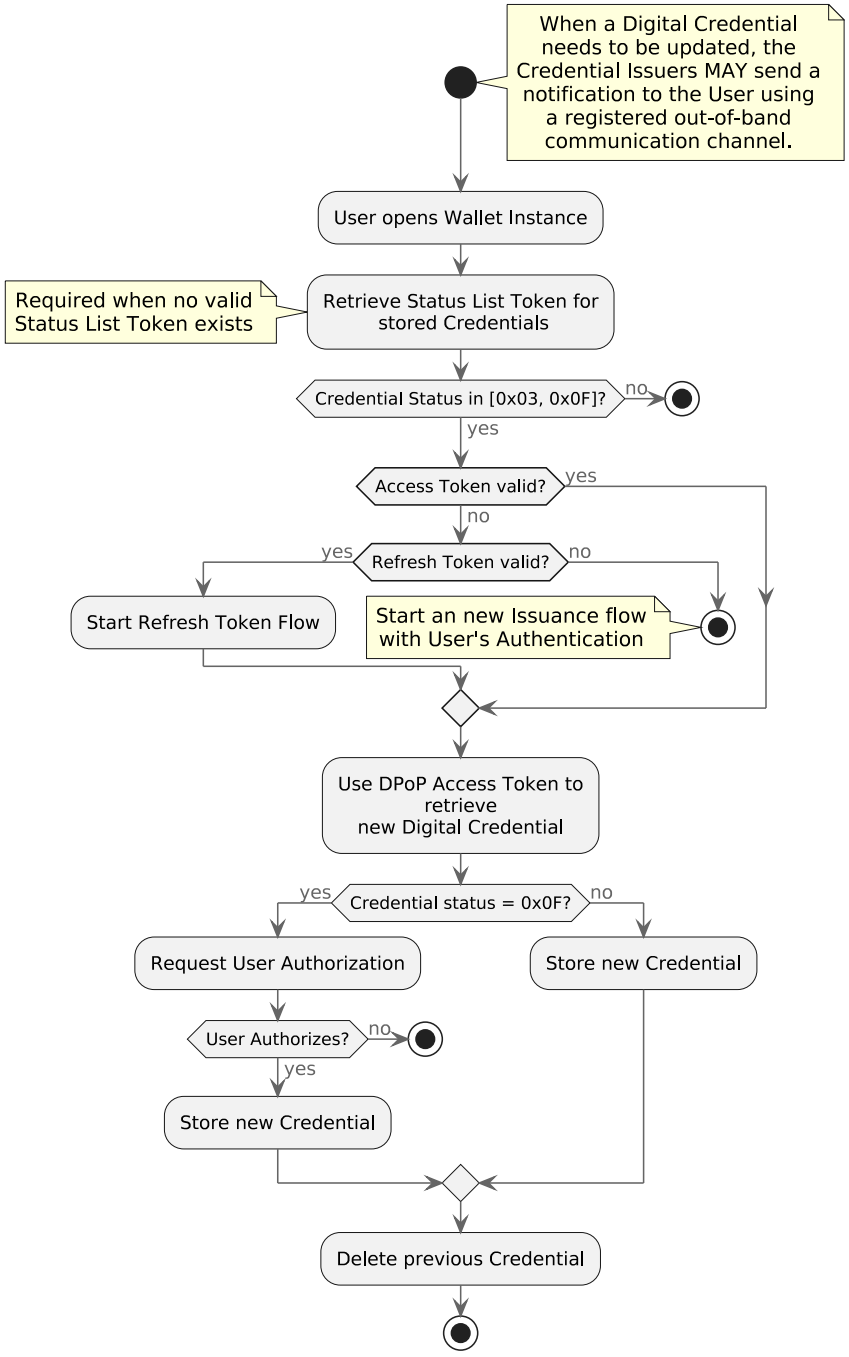


Fig. 11.2: Gestione del Ciclo di Vita degli Attestati Elettronici.

- Il ciclo di vita della sua Istanza del Wallet
- La validità degli Attributi gestiti dalle Fonti Autentiche
- Solo per i PID, lo stato dell'Identità Digitale utilizzata per l'autenticazione dell'Utente

Fattori esterni relativi all'utente possono anche influenzare il ciclo di vita di un Attestato Elettronico, come:

- Richiesta esplicita dal titolare dell'Attestato Elettronico
- Morte dell'Utente
- Attività illegali

Entità Coinvolte

Mentre il Fornitore di Attestati Elettronici DEVE gestire direttamente lo stato di validità degli Attestati Elettronici che ha emesso, altri attori POSSONO attivare il processo di revoca/sospensione dell'Attestato Elettronico:

- Utenti, attraverso:
  - La loro Istanza del Wallet
  - Servizio web fornito dal Fornitore di Attestati Elettronici
- La Fonte Autentica quando gli attributi dell'Attestato Elettronico vengono aggiornati o cambiano stato di validità
- Il Fornitore di Wallet quando revoca un'Istanza del Wallet
- Il Gestore di Identità Digitale se l'Identità Digitale utilizzata per il rilascio del PID/IT-Wallet ID è stata rubata o compromessa
- Autorità legali o l'Organo di Vigilanza in caso di attività illegali comprovate

La seguente figura mostra un diagramma delle relazioni tra entità relative al flusso di aggiornamento dello stato.

Flussi di Aggiornamento dello Stato

Questa sezione descrive i principali flussi per la gestione degli Aggiornamenti di Stato degli Attestati Elettronici da parte del Fornitore di Attestati Elettronici, in particolare l'Aggiornamento di Stato:

- relativo all'Utente;
- attivato da un'Istanza del Wallet;
- attivato da un Fornitore di Wallet;

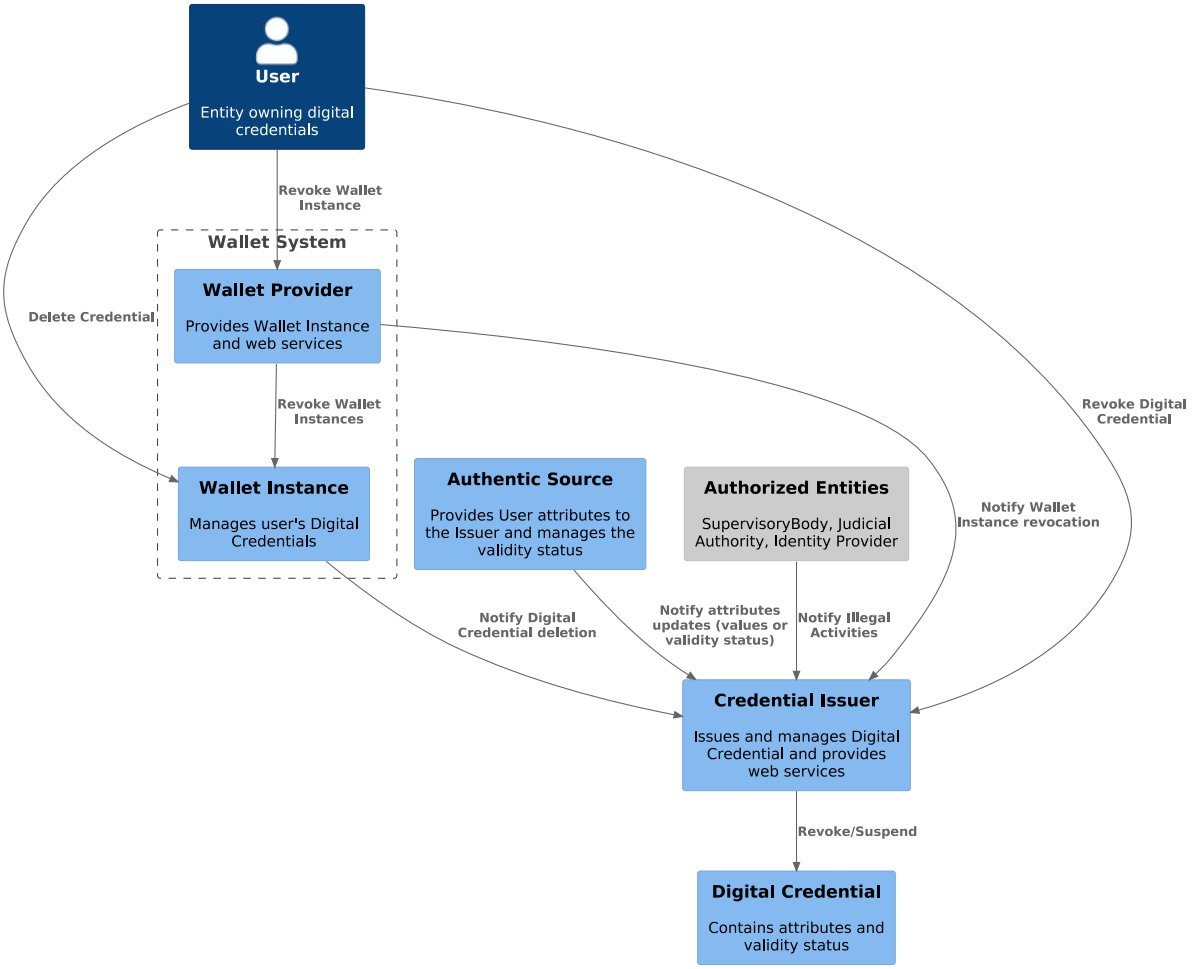


Fig. 11.3: Entità coinvolte nel Flusso di Revoca degli Attestati Elettronici.

- attivato da una Fonte Autentica.

Nota

I Flussi dettagliati di Aggiornamento dello Stato per i Gestori di Identità Digitale, le autorità legali e l'Organo di Vigilanza saranno trattati nelle versioni future della specifica tecnica.

Aggiornamento dello Stato relativo all'Utente

Gli Utenti POSSONO modificare lo stato di validità del loro Attestato Elettronico:

1. Eliminando l'Attestato Elettronico dalla propria Istanza del Wallet: l'Istanza del Wallet DOVREBBE proporre all'Utente un prompt per notificare opzionalmente al Fornitore di Attestati Elettronici il desiderio da parte dell'Utente di richiedere la revoca della Credenziale. Quando l'Utente utilizza questa funzionalità, la notifica da inviare al Fornitore di Attestati Elettronici DEVE utilizzare il Notification Endpoint messo a disposizione dal Fornitore di Attestati Elettronici, come descritto nella Sezione *Aggiornamento dello Stato da parte dell'Istanza del Wallet* (pagina 431).
2. Utilizzando il portale web del Fornitore di Attestati Elettronici:
   a. Gli Utenti POSSONO accedere ad un'area sicura con almeno lo stesso Livello di Garanzia utilizzato durante la fase di emissione.
   b. Il Fornitore di Attestati Elettronici DEVE consentire agli Utenti di:
   - Visualizzare tutti i loro Attestati Elettronici contenuti nel database del Fornitore di Attestati Elettronici.
   - Verificare l'autenticità dei dati.
   - Visualizzare e aggiornare lo stato di validità (revocare i loro Attestati Elettronici e, se supportato dal Fornitore di Attestati Elettronici, sospenderle).

Inoltre, quando gli Utenti rilevano dati non corretti in un Attestato Elettronico rilasciato, l'Istanza del Wallet DOVREBBE avviare una richiesta di correzione dati tramite il Notification Endpoint come specificato in credential-issuance-endpoint:Correzione dati usando credential_failure. A seguito della conferma della discrepanza, il Fornitore di Attestati Elettronici DOVREBBE seguire il *Re-issuance Flow* (pagina 473).

Nota

Se l'Utente attiva un'altra Istanza del Wallet dello stesso Fornitore di Wallet e utilizzando la stessa Soluzione di Wallet e ottiene un nuovo PID/IT-Wallet ID, il precedente PID/IT-Wallet ID DEVE essere revocato. In caso di revoca del PID la precedente Istanza del Wallet DEVE passare allo stato operativo.

In caso di morte dell'Utente, i Fornitori di Attestati Elettronici DEVONO garantire che gli Attestati Elettronici e le Istanze del Wallet di proprietà dell'Utente siano revocati. La morte dell'Utente comporta una modifica dello stato di validità degli attributi identificativi dell'Utente contenuti nel registro pubblico (ANPR). La morte dell'Utente DEVE produrre la revoca del PID/IT-Wallet ID. Pertanto, la Fonte Autentica del PID/IT-Wallet ID (ANPR) DEVE notificare al PID/EAA Provider che gli attributi dell'Utente non sono più validi a causa della morte dell'Utente. La Fonte Autentica e il PID/EAA Provider DEVONO utilizzare i meccanismi previsti nella Sezione *Aggiornamento dello Stato da parte delle Fonti Autentiche* (pagina 432).

Nota

Le versioni future della presente specifica tecnica definiranno come le informazioni verso i Fornitori di (Q)EAA vengono propagate, in accordo alla normativa nazionale. Inoltre, saranno definite procedure automatizzate per la revoca degli Attestati dovuta ad attività illecite.

Aggiornamento dello Stato da parte dell'Istanza del Wallet

Quando l'Utente elimina un Attestato Elettronico dall'Istanza del Wallet, per impostazione predefinita l'Istanza del Wallet NON DEVE notificare questo evento al Fornitore di Attestati Elettronici. L'eliminazione dall'Istanza del Wallet rimuove solo la copia locale e non modifica lo stato di validità presso il Fornitore di Attestati Elettronici.

L'Istanza del Wallet PUÒ informare l'Utente, prima dell'eliminazione, che l'eliminazione è un'azione locale e non implica la revoca presso il Fornitore di Attestati Elettronici, e PUÒ implementare, con il consenso esplicito dell'Utente al momento dell'eliminazione, una funzionalità di notifica per informare il Fornitore di Attestati Elettronici dell'intenzione dell'Utente di revocare l'Attestato Elettronico.

Se l'Utente desidera che il Fornitore di Attestati Elettronici revochi un Attestato Elettronico, DOVREBBE confermare esplicitamente tale intenzione tramite il prompt di eliminazione dell'Istanza del Wallet (quando disponibile), che a sua volta DEVE notificare il Fornitore di Attestati Elettronici; in alternativa, l'Utente PUÒ utilizzare il portale web del Fornitore di Attestati Elettronici o altri canali messi a disposizione.

Quando l'Attestato Elettronico revocato è il PID/IT-Wallet ID, il Fornitore di Attestati Elettronici DEVE inviare una notifica di questo evento all'Utente entro 24 ore. Per qualsiasi

altro Attestato Elettronico diverso dal PID/IT-Wallet ID, il Fornitore di Attestati Elettronici DOVREBBE inviare una notifica di questo evento all'Utente. La notifica all'Utente PUÒ essere implementata in diversi modi, ad esempio utilizzando l'indirizzo email dell'Utente, il numero di telefono o qualsiasi altro canale di comunicazione verificato e sicuro. La notifica all'Utente DEVE includere anche tutte le informazioni sullo stato di revoca dell'Attestato Elettronico. Il metodo utilizzato per la notifica all'Utente è fuori dallo scopo del presente profilo tecnico di implementazione. Quando la revoca avviene, il Fornitore di Attestati Elettronici DEVE aggiornare di conseguenza lo stato dell'Attestato Elettronico. Quando la Notification Response inviata dal Fornitore di Attestati Elettronici viene ricevuta con successo dall'Istanza del Wallet, l'Istanza del Wallet DEVE eliminare l'Attestato Elettronico.

Aggiornamento dello Stato da parte dei Fornitori di Wallet

Il Fornitore di Wallet che, per qualsiasi motivo, revoca una Wallet Instance DEVE garantire che lo stato aggiornato sia riportato nella status list della relativa Wallet Unit Attestation. In aggiunta a quanto già definito in *Ciclo di Vita degli Attestati Elettronici* (pagina 422), il Credential Issuer DEVE implementare un meccanismo di monitoraggio degli stati correnti di tutte le Wallet Unit Attestation relative alle Wallet Instance a cui sono stati emessi gli Attestati Elettronici. Dopo la revoca della Wallet Unit Attestation, il Credential Issuer procede alla revoca di tutti gli Attestati Elettronici emessi per la Wallet Instance.

Aggiornamento dello Stato da parte delle Fonti Autentiche

Le Fonti Autentiche gestiscono gli attributi separatamente dagli Attestati Elettronici, i quali verificano l'autenticità in modo analogo ai documenti fisici. La perdita di un documento fisico non implica la perdita dei diritti che rappresenta; significa solo che l'Utente non può dimostrarli. Tuttavia, se un Utente perde determinati diritti a causa di una grave infrazione, la Fonte Autentica revocherà gli attributi correlati. In tali casi, quando gli attributi di un Utente vengono aggiornati, le Fonti Autentiche DEVONO notificare ai Fornitori di Attestati Elettronici di aggiornare lo stato di validità di qualsiasi Attestato Elettronico che contenga tali attributi.

Le Fonti Autentiche che utilizzano PDND DEVONO usare Signal Hub come unico meccanismo di notifica degli aggiornamenti. In questo caso, le Fonti Autentiche DEVONO depositare un Signal tramite il *e-Service di Raccolta Segnali* (pagina 640) nei seguenti casi:

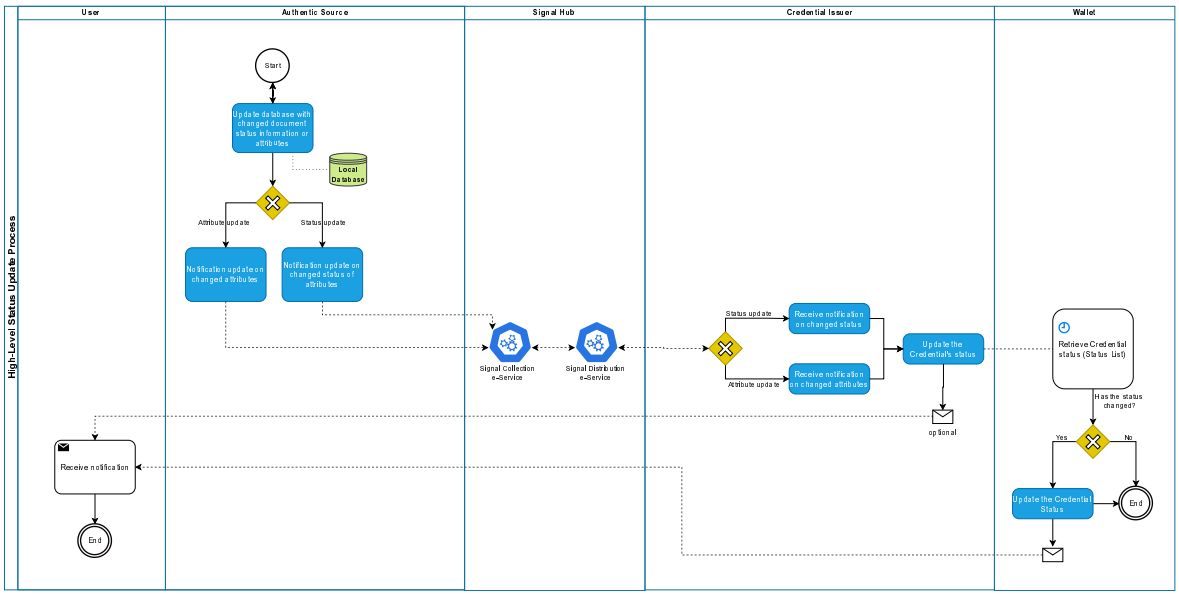
- Il valore di uno o più Attributi contenuti nel database della Fonte Autentica è cambiato;
- Lo stato di validità degli Attributi è aggiornato (revoca o sospensione).

In entrambi i casi, il Signal DEVE avere `signalType` impostato a `UPDATE`.

I Fornitori di Attestati Elettronici DEVONO controllare periodicamente il PDND Signal Hub *e-Service di Distribuzione Segnali* (pagina 642) per nuovi Signal. Per il flusso di processamento

dei Signal, fare riferimento alla Sezione *Elaborazione dei Segnali* (pagina 644). Il Fornitore di Attestati Elettronici è in grado di identificare la natura del `UPDATE` Signal interrogando la API *get attribute* della Fonte Autentica e ispezionando il payload della risposta, come descritto nella Sezione *Get Attribute Claims* (pagina 634).

La seguente figura illustra ad alto livello il processo di aggiornamento dello stato da parte delle Fonti Autentiche.



Il processo inizia con modifiche ai dati o alla validità dei dati che avvengono presso la Fonte Autentica. Le modifiche possono anche essere avviate da entità terze diverse dalla Fonte Autentica, ad esempio quando le autorità giudiziarie segnalano attività illecite.

Una volta che i dati cambiano, la Fonte Autentica notifica i Fornitori di Attestati Elettronici che hanno ricevuto i dati originali utilizzando il Signal Hub. La Fonte Autentica deposita un Signal nel Signal Collection e-Service. *e-Service di Raccolta Segnali* (pagina 640).

Il Fornitore di Attestati Elettronici interroga periodicamente il Signal Hub *e-Service di Distribuzione Segnali* (pagina 642) per nuovi Signal. Quando viene trovato un nuovo Signal, il Fornitore di Attestati Elettronici lo recupera e lo processa come descritto in *Elaborazione dei Segnali* (pagina 644). Quindi, il Fornitore di Attestati Elettronici aggiorna lo Stato dell'Attestato secondo la modalità definita dal meccanismo di validità. Il Fornitore di Attestati Elettronici PUÒ notificare l'Utente tramite un canale di comunicazione registrato e out-of-band.

L'Istanza del Wallet, a seguito di controlli periodici dello stato di validità degli Attestati Elettronici memorizzati, riceve lo stato aggiornato. Quando lo Stato dell'Attestato viene modificato in `INVALID`, il Fornitore di Attestati Elettronici DEVE informare l'Utente di tale cambiamento. Nel caso in cui lo stato dell'Attestato venga modificato in `UPDATE` (risp. 0x03) o `ATTRIBUTE_UPDATE` (risp. 0x0F), l'Istanza del Wallet DOVREBBE procedere alla riemissione dell'Attestato Elettronico, come descritto in *Re-issuance Flow* (pagina 473).

Gestione del ciclo di vita delle Credenziali in batch

Quando più Attestati Elettronici vengono emessi insieme in un singolo batch, il loro ciclo di vita rimane completamente granulare:

- **Trigger raggruppati, aggiornamenti indipendenti**: indipendentemente dall'attore che ha innescato la variazione di stato del batch (ad esempio, la Wallet Instance tramite il Notification Endpoint con `event=credential_deleted`, un Wallet Provider tramite l'aggiornamento della Status List della Wallet Instance e Key Attestations) l'aggiornamento viene gestito come una o più modifiche di stato separate. Il Fornitore di Attestati Elettronici aggiorna lo stato di ciascun Attestato Elettronico singolarmente (ad esempio, impostando il bit della status-list su `INVALID` o `SUSPENDED`). Un'Istanza del Wallet NON DEVE attivare aggiornamenti di stato del batch quando l'Utente elimina Attestati Elettronici localmente. In fase di eliminazione, l'Istanza del Wallet PUÒ, con il consenso esplicito dell'Utente, notificare al Fornitore di Attestati Elettronici l'intenzione dell'Utente di revocare gli Attestati Elettronici interessati.

Nota

Poiché l'interfaccia utente del Wallet in genere visualizza un batch come un singolo Attestato Elettronico (ad esempio, con 3 utilizzi rimanenti), un'eliminazione da parte dell'utente rimuove l'intero batch localmente. Per impostazione predefinita non richiede la revoca presso il Fornitore di Attestati Elettronici. L'Istanza del Wallet PUÒ offrire all'Utente un prompt opzionale per richiedere la revoca presso il Fornitore di Attestati Elettronici nell'ambito del flusso di eliminazione.

11.4.4 Meccanismi di Verifica della Validità

Per la verifica dello stato di validità di un Attestato Elettronico a lunga durata, Token Status List (TOKEN-STATUS-LIST) DEVE essere supportato sia per lo scenario remoto che per quello di prossimità. La seguente tabella riassume i meccanismi di revoca richiesti per verificare lo stato degli Attestati Elettronici a lunga durata.

Flusso	Meccanismo di Revoca	Riferimento
Remoto	• [OBBLIGATORIO] Token Status List.	TOKEN-STATUS-LIST.
Prossimità	[OBBLIGATORIO] Token Status List.	TOKEN-STATUS-LIST.

Token Status Lists

Questa sezione definisce una struttura di dati Status List, che viene utilizzata per trasmettere informazioni riguardanti gli stati individuali di più Attestati Elettronici. Gli Attestati Elettronici possono essere di qualsiasi formato, come SD-JWT o mdoc ISO/IEC 18013-5. Una Status List descrive lo stato degli Attestati Elettronici codificando la loro validità in un array di bit. A ciascun Attestato Elettronico viene assegnato un indice durante l'emissione; questo indice rappresenta la sua posizione all'interno dell'array di bit. Il valore del bit o dei bit in questo indice corrisponde allo stato degli Attestati Elettronici. Una Status List viene fornita all'interno di un Token di Status List firmato crittograficamente in formato JWT. Per i dettagli, vedere TOKEN-STATUS-LIST.

In questa specifica, i ruoli di Fornitore di Attestati Elettronici e Status Issuer (cioè, l'entità che emette il Token di Status List sulle informazioni di stato dell'Attestato Elettronico) coincidono, mentre lo Status Provider (cioè, l'entità che fornisce il Token di Status List su un endpoint pubblico) PUÒ essere il Fornitore di Attestati Elettronici stesso o un'altra entità.

Il Fornitore di Attestati Elettronici DEVE

- definire un numero di bit, k, (1, 2, 4 o 8) che rappresenta la quantità di bit utilizzati per descrivere lo stato di ciascun Attestato Elettronico all'interno di questa Status List. Il Fornitore di Attestati Elettronici DEVE configurare il numero di bit. Ogni Attestato Elettronico avrà quindi 2^k (dove k è il numero di bit scelto) possibili stati.
- creare un array di byte di dimensione = (quantità di Attestati Elettronici) * k / 8 o maggiore. A seconda di k, ogni byte nell'array corrisponde a 8/k stati (8 se k=1, 4 se k=2, 2 se k=1, o 1 se k=8). Ogni volta che viene emesso un Attestato Elettronico, il Fornitore di Attestati Elettronici lo assegna a una posizione nell'array.
- impostare i valori di stato per tutti gli Attestati Elettronici emessi all'interno dell'array di byte. Lo stato di ciascun Attestato Elettronico viene identificato utilizzando un indice che mappa a uno o più bit specifici all'interno dell'array di byte. L'indice inizia a contare da 0 e termina con (quantità di Attestati Elettronici) - 1 (essendo l'ultima voce valida). I bit all'interno di un array vengono contati dal bit meno significativo ("0") al bit più significativo ("7"). Tutti i bit dell'array di byte a un particolare indice sono impostati su un valore di stato.
- comprimere l'array di byte utilizzando DEFLATE [**RFC 1951**] con il formato dati ZLIB [**RFC 1950**]. Si RACCOMANDA alle implementazioni di utilizzare il livello di compressione più alto disponibile.
- rendere disponibile alle Relying Party e alle Istanze del Wallet un endpoint per richiedere le Status Lists.

Il Fornitore di Attestati Elettronici DEVE utilizzare i seguenti valori per i possibili Stati degli Attestati Elettronici emessi:

- 0x00 - `VALID` - L'Attestato Elettronico è valido.
- 0x01 - `INVALID` - L'Attestato Elettronico è revocato.
- 0x02 - `SUSPENDED` - L'Attestato Elettronico è temporaneamente non valido, sospeso. Questo stato è solitamente temporaneo.
- 0x03 - `UPDATE` - I parametri dei metadata dell'Attestato Elettronico sono cambiati.
- 0x0F - `ATTRIBUTE_UPDATE` - Gli attributi dell'Attestato Elettronico sono cambiati.

Ad esempio, se sono possibili cinque stati per un certo Attestato Elettronico, allora k=4. Se il Fornitore di Attestati Elettronici crea un array per memorizzare gli stati di 6 Attestati Elettronici, i cui stati di validità sono 0, 0, 0, 3, 1, 2, rispettivamente; farà:

- creare l'array di bit `[0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 1, 1, 0, 0, 0, 0; 0, 0, 1, 0, 0, 0, 0, 1]` che in notazione esadecimale genera l'array di byte `[0x00, 0x30, 0x21]`.
- comprimere l'array utilizzando DEFLATE.

Nota

Quando il Fornitore di Attestati Elettronici sceglie il numero di bit per trasmettere gli stati degli Attestati Elettronici che emette, PUÒ aggiungere altri stati oltre a quelli descritti sopra. L'aggiunta di molti stati diversi per il ciclo di vita di un Attestato Elettronico deve tuttavia essere attentamente ponderata poiché rivela informazioni alle Relying Party.

Nota

La principale considerazione sulla privacy per una Status List è impedire al Fornitore di Attestati Elettronici di tracciare l'uso dell'Attestato Elettronico quando lo stato viene controllato. Se un Fornitore di Attestati Elettronici offre informazioni sullo stato facendo riferimento a un token specifico, ciò consentirebbe al Fornitore di Attestati Elettronici di creare un profilo per il token emesso correlando la data e l'identità delle Relying Party che richiedono lo stato. Le implementazioni DEVONO quindi integrare le informazioni sullo stato di molti Attestati Elettronici nella stessa lista. Di conseguenza, il Fornitore di Attestati Elettronici non apprende per quale Attestato Elettronico la Relying Party sta richiedendo la Status List. La privacy dell'Utente è protetta dall'anonimato all'interno dell'insieme degli Attestati Elettronici nella Status List, questo limita le possibilità di tracciamento da parte del Fornitore di Attestati Elettronici. Questo effetto di privacy di gruppo dipende dal numero di entità all'interno della Status List. Una maggiore quantità di Attestati Elettronici referenziati in essa risulta in una migliore privacy ma influisce anche sulle prestazioni poiché più dati

devono essere trasferiti per leggere la Status List. A seconda dei parametri della Status List (ad esempio la quantità di bit che designano i valori dell'Attestato Elettronico), i Fornitori di Attestati Elettronici devono trovare un equilibrio appropriato tra privacy e prestazioni.

Una volta che la Relying Party riceve un Attestato Elettronico, questo le consente di richiedere la Status List per validare il suo stato attraverso il parametro URI fornito e cercare l'indice corrispondente. Tuttavia, la Relying Party è in grado di memorizzare l'URI e l'indice dell'Attestato Elettronico per richiedere nuovamente la Status List in un momento successivo. Facendolo regolarmente, la Relying Party può creare un profilo dello stato di validità dell'Attestato Elettronico. Questo comportamento potrebbe anche essere abusato in casi in cui non è intenzionale e sconosciuto all'Utente, ad esempio profilando la sospensione di una patente di guida. Questo comportamento potrebbe essere mitigato, ad esempio, con la regolare riemissione dell'Attestato Elettronico.

Token di Status List

Il Token di Status List è disponibile all'Endpoint di Status List e contiene i seguenti parametri.

Header	Descrizione	Riferimento
alg	OBBLIGATORIO. Un identificatore di algoritmo di firma digitale come per il registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` o su un identificatore di algoritmo simmetrico (MAC).	[**RFC 7515**], [**RFC 7517**].
typ	OBBLIGATORIO. DEVE corrispondere al valore `statuslist+jwt`.	TOKEN-STATUS-LIST
kid	OBBLIGATORIO. Identificatore univoco della chiave pubblica del Fornitore di Attestati Elettronici che firma il Token di Status.	**RFC 7638# section_3**.
x5c	OBBLIGATORIO. Certificato di chiave pubblica X.509 o catena di certificati corrispondente alla chiave utilizzata per firmare il Token di Status List.	**RFC 5280**

Payload	Descrizione	Riferimento
sub	OBBLIGATORIO. Il claim del soggetto DEVE specificare l'URI del Token di Status List. Il valore DEVE essere uguale a quello del claim `uri` contenuto nel claim `status_list` dell'Attestato Elettronico.	[**RFC 7519**]

continues on next page

Tabella 11.18 – continua dalla pagina precedente

Payload	Descrizione	Riferimento
iat	OBBLIGATORIO. Il claim issued at DEVE specificare l'ora in cui è stato emesso il Token di Status List.	[**RFC 7519**]
exp	RACCOMANDATO. Il claim expiration time, se presente, DEVE specificare l'ora in cui il Token di Status List è considerato scaduto dal Fornitore di Attestati Elettronici.	[**RFC 7519**]
ttl	RACCOMANDATO. Il claim time to live, se presente, DEVE specificare la quantità massima di tempo, in secondi, che il Token di Status List può essere memorizzato nella cache da un consumatore prima che una copia aggiornata DOVREBBE essere recuperata. Il valore del claim DEVE essere un numero positivo codificato in JSON come un numero. Questa quantità di tempo NON DOVREBBE superare il tempo di scadenza definito nel claim **exp**.	TOKEN-STATUS-LIST
status_list	OBBLIGATORIO. Oggetto JSON che contiene una Status List.	TOKEN-STATUS-LIST

Nota

Si RACCOMANDA che il Fornitore di Attestati Elettronici imposti il claim `exp` in modo che il Token di Status List abbia una breve durata. In genere, questo comporta che il claim `exp` non superi il claim `iat` di più di 24 ore.

Una Status List codificata in JSON ha la seguente struttura:

Parametro	Descrizione	Riferimento
bits	OBBLIGATORIO. Intero JSON che specifica il numero di bit per Attestato Elettronico nell'array di byte compresso (*lst*). I valori consentiti per bits sono 1,2,4 e 8.	TOKEN-STATUS-LIST
lst	OBBLIGATORIO. Stringa JSON che contiene i valori di stato per tutti gli Attestato Elettronici di cui trasmette gli stati. Il valore DEVE essere l'array di byte compresso codificato in base64url.	TOKEN-STATUS-LIST
aggregation_uri	OPZIONALE. Stringa JSON che contiene un URI per recuperare l'Aggregazione di Status List per questo tipo di Attestato Elettronico o Fornitore di Attestati Elettronici.	TOKEN-STATUS-LIST

Di seguito è riportato un esempio di Token di Status List prima di applicare la firma e la codifica:

```
{
  "alg": "ES256",
  "kid": "$KID",
```

```
  "typ": "statuslist+jwt",
  "x5c": [
    "MIIDqjCCApKgAwIBAgIESLNEvDA ...",
    "MIICwzCCAasCCQCKVy9eKjvi+jA ...",
    "MIIDTDCCAjSgAwIBAgIJAPlnQYH..."
  ]
}
```

```
{
  "exp": 2291720170,
  "iat": 1686920170,
  "status_list": {
    "bits": 1,
    "lst": "eNrbuRgAAhcBXQ"
  },
  "sub": "https://example-issuer.com/statuslists/",
  "ttl": 43200
}
```

Gestione dello Stato degli Attestati Elettronici con Token di Status List

I Fornitori di Attestati Elettronici, una volta che un Attestato Elettronico è stato generato, DEVONO:

- Memorizzarlo localmente con il set minimo di dati necessari per gestire il suo ciclo di vita, incluso lo stato di validità di quell'Attestato Elettronico;
- Includere un claim `status_list` all'interno dell'Oggetto JSON `status` claim dell'Attestato Elettronico.

Il valore del claim `status_list` DEVE essere a sua volta un Oggetto JSON con i seguenti parametri

Parametro	Descrizione	Riferimento
idx	OBBLIGATORIO. Il claim idx (indice) DEVE specificare un Intero che rappresenta l'indice da controllare per le informazioni sullo stato nella Status List per l'Attestato Elettronico corrente. Il valore di idx DEVE essere un numero non negativo, contenente il valore zero o maggiore.	TOKEN-STATUS-LIST
uri	OBBLIGATORIO. Il claim uri (URI) DEVE specificare un valore String che identifica il Token di Status List contenente le informazioni sullo stato per l'Attestato Elettronico. Il valore di uri DEVE essere un URI conforme a [**RFC 3986**].	TOKEN-STATUS-LIST

Verifica degli Stati degli Attestati Elettronici

Il recupero, l'elaborazione e la verifica di un Token di Status List possono essere effettuati sia dall'Istanza del Wallet che da una Relying Party. Di seguito viene descritto per la Wallet Instance, tuttavia, le stesse regole si applicherebbero anche al Relying Party.

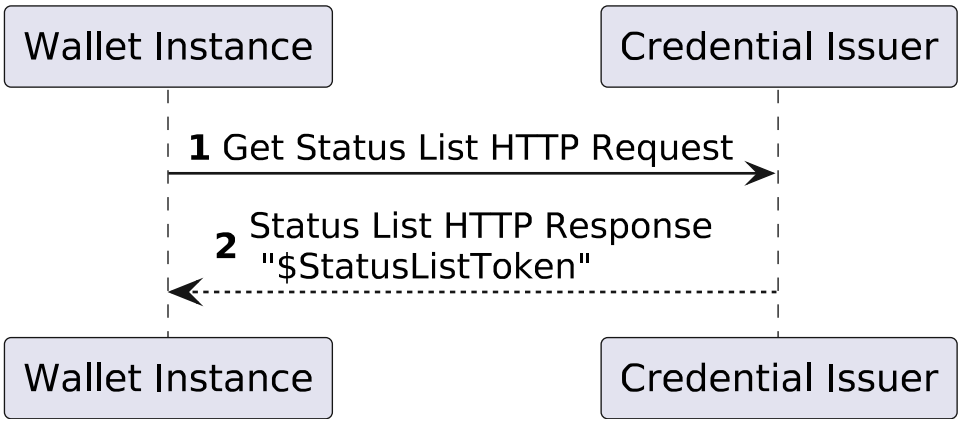


Fig. 11.4: Flusso di Status List.

Richiesta HTTP di Status List

Per ottenere il Token di Status List, la Relying Party DEVE inviare una richiesta HTTP GET al valore `status.status_list.uri` fornito all'interno dell'Attestato Elettronico.

La Relying Party DOVREBBE inviare l'Accept-Header `application/statuslist+jwt` per indicare che il tipo di risposta richiesto per il Token di Status List è il formato JWT.

Di seguito è riportato un esempio non normativo di una richiesta per un Token di Status List:

```
GET /statuslists HTTP/1.1
Host: example-issuer.com
Accept: application/statuslist+jwt
```

Risposta HTTP di Status List

L'Endpoint di Status List risponde con un Token di Status List e DEVE utilizzare un codice di stato HTTP nell'intervallo 2xx. In caso di risposta andata a buon fine, lo Status Provider DEVE utilizzare il content-type `application/statuslist+jwt` per il Token di Status List in formato JWT.

La risposta HTTP DOVREBBE utilizzare la Content-Encoding gzip come definito in [RFC 9110].

Se gli header HTTP relativi alla cache sono presenti nella risposta HTTP, le Relying Party DOVREBBERO dare priorità ai claim `exp` e `ttl` all'interno del Token di Status List rispetto agli header HTTP per determinare il comportamento della cache.

Di seguito è riportato un esempio non normativo di una risposta per un Token di Status List con tipo `application/statuslist+jwt`:

```
HTTP/1.1 200 OK
Content-Type: application/statuslist+jwt

eyJhbGciOiJFUzI1NiIsImtpZCI6IjEyIiwidHlwIjoic3RhdHVzbGlzdCtqd3QifQ.eyJleHAiOjIyOTE3MjAxNzAsIml
hdCI6MTY4NjkyMDE3MCwiaXNzIjoiaHR0cHM6Ly9leGFtcGxlLmNvbSIsInN0YXR1c19saXN0Ijp7ImJpdHMiOjEsImxzd
CI6ImVOcmJ1UmdBQWhjQlhRIn0sInN1YiI6Imh0dHBzOi8vZXhhbXBsZS5jb20vc3RhdHVzbGlzdHMvMSIsInR0bCI6NDM
yMDB9.SSdg3AnTHsyRtCHziLy-QnXg-YRldMEXkdEgDXgE_ZvIvjM0eULQlzEbLBLfCeGhlqKJSReC-m85K79CTjJDzg
```

Al ricevimento di un Attestato Elettronico, una Relying Party DEVE prima eseguire la validazione dell'Attestato Elettronico stesso (ad esempio, controllando gli attributi previsti, la firma valida e il tempo di scadenza). Se questa validazione non ha successo, l'Attestato Elettronico DEVE essere rifiutata. Se la validazione ha avuto successo, la Relying Party DEVE eseguire i seguenti passaggi di validazione per valutare lo stato dell'Attestato Elettronico:

- Controllare l'esistenza di un claim `status`, controllare l'esistenza di un claim `status_list` all'interno del claim `status` e validare che il contenuto di `status_list` aderisca alle regole definite nella Sezione *Gestione dello Stato degli Attestati Elettronici con Token di Status List* (pagina 439).
- Risolvere il Token di Status List dall'URI fornito.
- Validare il Token di Status List:
  - Validare la firma del Token di Status List seguendo le regole definite nella sezione 7.2 di [RFC 7519]. Questo passaggio richiede la risoluzione di una chiave pubblica come descritto in *L'Infrastruttura di Trust* (pagina 117).
  - Controllare l'esistenza dei claim richiesti come definito nella Sezione *Token di Status List* (pagina 437).
- Tutti i claim esistenti nel Token di Status List DEVONO essere controllati secondo *Token di Status List* (pagina 437).
  - Il claim subject del Token di Status List DEVE essere uguale al claim `` `uri` `` nell'oggetto `status_list` dell'Attestato Elettronico.
  - Se la Relying Party ha politiche personalizzate riguardanti la freschezza del Token di Status List, DOVREBBE controllare il claim `iat`.

  - Se il tempo di scadenza è definito, DEVE essere controllato se il Token di Status List è scaduto.
  - Se la Relying Party sta utilizzando un sistema per memorizzare nella cache il Token di Status List, DOVREBBE controllare il claim `ttl` del Token di Status List e recuperare una copia aggiornata se (tempo in cui lo stato è stato risolto + *ttl* < tempo corrente).

- Decomprimere la Status List con un decompressore compatibile con DEFLATE [**RFC 1951**] e ZLIB [**RFC 1950**].
- Recuperare il valore di stato dell'indice specificato nell'Attestato Elettronico come descritto in *Verifica degli Stati degli Attestati Elettronici* (pagina 440). Fallire se l'indice fornito è fuori dai limiti della Status List.
- Controllare il valore di stato come descritto in *Verifica degli Stati degli Attestati Elettronici* (pagina 440).

Se uno di questi controlli fallisce, non è possibile fare alcuna dichiarazione sullo stato dell'Attestato Elettronico e l'Attestato Elettronico DOVREBBE essere rifiutato.

Se, ad esempio, l'array di byte decompresso è `[0x00, 0x40, 0x21]`, corrisponde all'array di bit `[0, 0, 0, 0, 0, 0, 0, 0; 0, 1, 0, 0, 0, 0, 0, 0; 0, 0, 1, 0, 0, 0, 0, 1]`. Lo Stato dell'Attestato Elettronico il cui valore del claim `idx` è `5` in questo array si riferisce all'ultima coppia di 4 bit (cioè, `[0, 0, 1, 0] = 0x02`) il cui valore di stato è `SUSPENDED`.

In caso di errore durante la generazione della risposta da parte dell'Endpoint del Token di Status, DEVONO essere supportati i seguenti Codici di Stato HTTP:

Codice di Stato	Descrizione
500 Internal Server Error [OBBLIGATORIO]	Il Provider di Status List ha riscontrato un problema interno.
503 Service Unavailable [OBBLIGATORIO]	Il Provider di Status List è temporaneamente non disponibile.
504 Gateway Timeout [OPZIONALE]	Il Provider di Status List non può soddisfare la richiesta entro l'intervallo di tempo definito.

CAPITOLO 12

Flussi relativi agli Attestati Elettronici

12.1 Emissione di Attestati Elettronici

Questa sezione descrive il flusso di emissione di un Attestato Elettronico di Dati di Identificazione Personale (PID) e di Attestati Elettronici di Attributi ((Q)EAA) con un elevato livello di sicurezza. Nelle sezioni successive vengono utilizzati i termini Credential Issuer, PID Provider, (Q)EAA Provider per riferirsi rispettivamente a Fornitore di Attestati Elettronici, Fornitore di Attestati Elettronici di Dati di Identificazione Personale e Fornitore di Attestati Elettronici di Attributi.

12.1.1 Flussi ad Alto Livello per l'Emissione di Attestati Elettronici

Flusso ad Alto Livello per PID

La Fig. 12.1 mostra un'architettura generale ed evidenzia le principali operazioni coinvolte nell'emissione del PID.

Il flusso ad alto livello inizia con l'Utente che desidera ottenere un PID e avvia la propria Istanza del Wallet (Passo 0). Di seguito la descrizione dei passaggi rappresentati nell'immagine precedente:

1. **Individuazione e Trust del PID Provider**: l'Istanza del Wallet individua il PID Provider fidato utilizzando il Catalogo degli Attestati Elettronici e i Servizi di Federazione, stabilendo la trust verso il PID Provider secondo il Trust Model e ottenendo i suoi

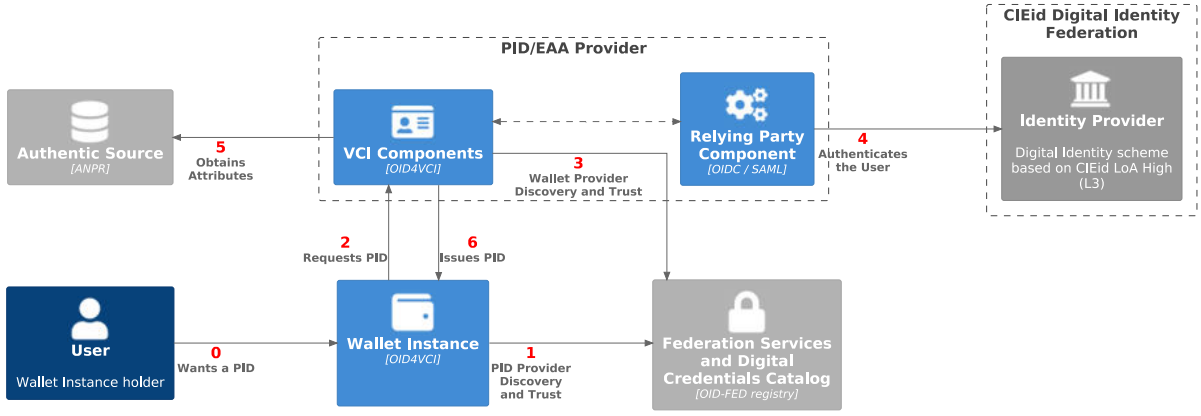


Fig. 12.1: Emissione del PID - Architettura generale e flusso ad alto livello.

Metadata, che indicano i formati del PID, gli algoritmi supportati e qualsiasi altro parametro necessario per esigenze di interoperabilità (*WP_045–046* (pagina 790)).

2. **Richiesta del PID**: utilizzando l'Authorization Code Flow definito in [OpenID4VCI], l'Istanza del Wallet richiede il PID al PID Provider (*WP_051* (pagina 790)).

3. **Individuazione e Trust del Fornitore di Wallet**: il PID Provider verifica l'autenticità e la validità dell'Istanza del Wallet, stabilendo la trust verso il Fornitore di Wallet e ottenendo i Metadata del Wallet con i parametri necessari per le esigenze di interoperabilità, secondo il Trust Model.

4. **Autenticazione dell'Utente**: il PID Provider autentica l'Utente secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.

5. **Recupero dei dati PID dal Registro Pubblico Nazionale**: il PID Provider ottiene i dati PID richiesti dal Registro Pubblico Nazionale (ANPR), che agisce come Fonte Autentica.

6. **Emissione del PID**: il PID Provider rilascia un PID vincolato al materiale crittografico posseduto dall'Istanza del Wallet richiedente.

Flusso ad Alto Livello per IT-Wallet ID

La Fig. 12.2 mostra un'architettura generale ed evidenzia le principali operazioni coinvolte nell'emissione di un IT-Wallet ID.

Il flusso ad alto livello inizia con l'Utente che desidera ottenere un IT-Wallet ID e avvia la propria Istanza del Wallet (Passo 0). Di seguito la descrizione dei passaggi rappresentati nell'immagine precedente:

1. **Individuazione e Trust del Provider di IT-Wallet ID**: l'Istanza del Wallet individua l'EAA Provider di IT-Wallet ID fidato utilizzando il Catalogo degli Attestati Elettronici e i Servizi di Federazione, stabilendo la trust verso l'EAA Provider di IT-Wallet secondo

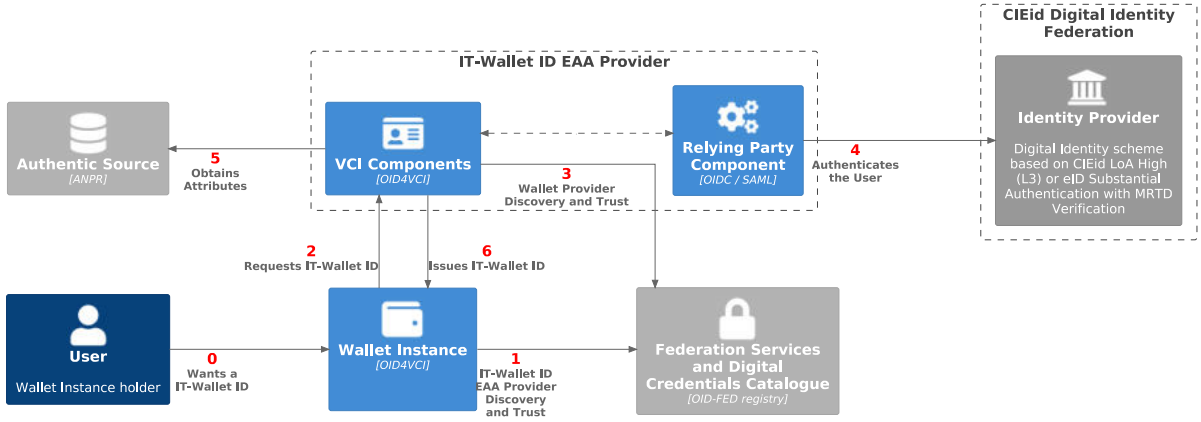


Fig. 12.2: Emissione dell'IT-Wallet ID - Architettura generale e flusso ad alto livello.

il Trust Model e ottenendo i suoi Metadata, che indicano i formati dell'IT-Wallet ID, gli algoritmi supportati e qualsiasi altro parametro necessario per esigenze di interoperabilità (*WP_045-046* (pagina 790)).

2. **Richiesta dell'IT-Wallet ID**: utilizzando l'Authorization Code Flow definito in [Open ID4VCI], l'Istanza del Wallet richiede l'IT-Wallet ID all'EAA Provider (*WP_051* (pagina 790)).

3. **Individuazione e Trust del Fornitore di Wallet**: l'EAA Provider di IT-Wallet ID verifica l'autenticità e la validità dell'Istanza del Wallet, stabilendo la trust verso il Fornitore di Wallet e ottenendo i Metadata del Wallet con i parametri necessari per le esigenze di interoperabilità, secondo il Trust Model.

4. **Autenticazione dell'Utente**: l'EAA Provider di IT-Wallet ID autentica l'Utente secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente. Per il flusso di protocollo L2+, vedere *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482).

5. **Recupero dei dati dell'IT-Wallet ID dal Registro Pubblico Nazionale**: l'EAA Provider di IT-Wallet ID ottiene i dati richiesti dell'IT-Wallet ID dal Registro Pubblico Nazionale (ANPR), che agisce come Fonte Autentica.

6. **Emissione dell'IT-Wallet ID**: l'EAA Provider di IT-Wallet ID rilascia un IT-Wallet ID vincolato al materiale crittografico posseduto dall'Istanza del Wallet richiedente.

Flusso ad Alto Livello per (Q)EAA

La Fig. 12.3 mostra un'architettura generale ed evidenzia le principali operazioni coinvolte nell'emissione di un (Q)EAA, seguendo le ipotesi elencate di seguito:

- l'Utente ha un PID valido memorizzato nella propria Istanza del Wallet, oppure un IT-Wallet ID valido quando richiesto secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente;
- il (Q)EAA richiede un profilo di implementazione ad alta sicurezza.

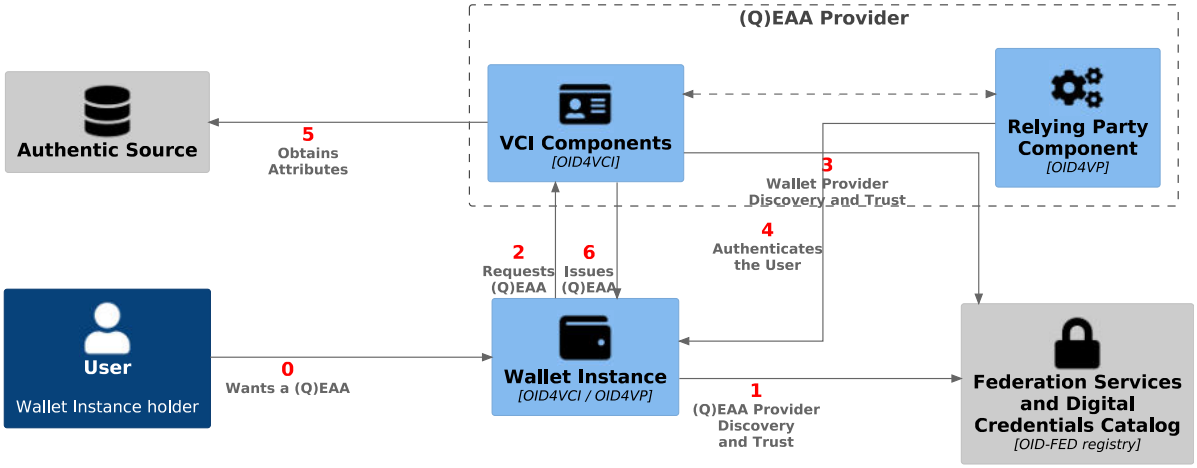


Fig. 12.3: Emissione di (Q)EAA - Architettura generale e flusso di alto livello.

Analogamente al flusso ad alto livello del PID e dell'IT-Wallet ID, il diagramma sopra illustra un flusso ad alto livello per (Q)EAA che inizia dall'Utente che desidera ottenere un (Q)EAA (passo 0). Di seguito la descrizione delle operazioni più rilevanti coinvolte nell'emissione del (Q)EAA:

1. **Individuazione e Trust del (Q)EAA Provider**: l'Istanza del Wallet ottiene l'elenco dei (Q)EAA Provider fidati utilizzando il Catalogo degli Attestati Elettronici e le API di Federazione (ad esempio, utilizzando l'endpoint Subordinate Listing del Trust Anchor e dei suoi Intermediari), quindi ispeziona i Metadata alla ricerca della disponibilità di Attestati Elettronici di ciascun (Q)EAA Provider (*WP_045–046* (pagina 790)).
2. **Richiesta del (Q)EAA**: utilizzando l'Authorization Code Flow, definito in [Open ID4VCI], l'Istanza del Wallet richiede un (Q)EAA al (Q)EAA Provider (*WP_051* (pagina 790)).
3. **Individuazione e Trust del Fornitore di Wallet**: il (Q)EAA Provider verifica l'autenticità e la validità dell'Istanza del Wallet. Durante questo passaggio, il (Q)EAA Provider stabilisce la trust con il Fornitore di Wallet e recupera i Metadata del Wallet contenenti i parametri necessari per l'interoperabilità, come definito dal Trust Model.

4. **Autenticazione dell'Utente**: il (Q)EAA Provider, agendo come App di Verifica, autentica l'Utente secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.
5. **Ottenimento degli Attributi**: il (Q)EAA Provider recupera gli Attributi dell'Utente dalla relativa Fonte Autentica.
6. **Emissione del (Q)EAA**: il (Q)EAA Provider rilascia un (Q)EAA vincolato al materiale crittografico posseduto dall'Istanza del Wallet richiedente.

12.1.2 Flussi Dettagliati per l'Emissione di Attestati Elettronici

Issuance Flow

Il flusso di emissione degli Attestati Elettronici (Issuance Flow) è basato su [OpenID4VCI] e i seguenti standard/specifiche di riferimento principali DEVONO essere supportati in aggiunta a OpenID4VCI:

- **The OAuth 2.0 Authorization Framework** [**RFC 6749**], come raccomandato nella Sezione 3 di [OpenID4VCI].
- **Pushed Authorization Requests** (PAR) [**RFC 9126**], come raccomandato nella Sezione 5 di [OpenID4VCI].
- **Proof Key for Code Exchange** (PKCE) [**RFC 7636**], come raccomandato nella Sezione 5 di [OpenID4VCI].
- **JWT Authorization Requests** (JAR) [**RFC 9101**].
- **Rich Authorization Requests** (RAR) [**RFC 9396**].
- **OAuth 2.0 Attestation-Based Client Authentication** [OAUTH-ATTESTATION-CLIENT-AUTH].
- **OpenID Federation 1.0** [OID-FED].

Il Credential Issuer DEVE utilizzare un *OAuth 2.0 Authorization Server* basato su **RFC 6749** per autorizzare l'Utente a ottenere un Attestato Elettronico. I Credential Issuer DEVONO supportare:

- **Authorization Code Flow**: Il Credential Issuer richiede l'autenticazione dell'Utente e il consenso all'Authorization Endpoint prima di raccogliere le informazioni dell'Utente per creare e rilasciare un Attestato Elettronico.
- **Wallet Initiated Flow**: La richiesta dell'Istanza del Wallet viene inviata al Credential Issuer senza alcun input dal Credential Issuer o una terza parte (i.e. tramite supporto al Credential Offer).

- **Immediate Issuance Flow**: Il Credential Issuer rilascia l'Attestato Elettronico direttamente in risposta alla Credential Request.

In aggiunta, i Credential Issuer POSSONO supportare:

- **Third-Party Initiated Flow**: L'Istanza del Wallet invia la sua richiesta al Credential Issuer in base all'input fornito dal Credential Issuer o da una terza parte (per esempio una Fonte Autentica) che supporta il meccanismo di Credential Offer definito in [Open ID4VCI]..
  - **Same-device Issuance Flow**: L'Utente riceve l'Attestato Elettronico sullo stesso dispositivo utilizzato per avviare il flusso.
  - **Cross-device Issuance Flow**: L'Utente riceve l'Attestato Elettronico su un dispositivo diverso da quello su cui ha avviato il flusso.
- **Refresh Token Flow**: L'Istanza del Wallet richiede un nuovo Access Token al Token Endpoint del PID/(Q)EEA Provider.
- **Re-issuance Flow**: A seguito di aggiornamenti ad un Attestato Elettronico già memorizzato, l'Istanza del Wallet richiede un aggiornamento dell'Attestato Elettronico al Credential Endpoint del Credential Issuer.
- **Deferred Issuance Flow**: Il Credential Issuer potrebbe impiegare del tempo per emettere l'Attestato Elettronico richiesto, a causa delle regole di provisioning dei dati delle Fonti Autentiche, e consente al Wallet di recuperare l'Attestato Elettronico richiesto in futuro.
- **Batch Credential Issuance Flow**: Permette l'emissione di un batch di uno o più Attestati Elettronici. Gli Attestati Elettronici emessi in batch DEVONO condividere lo stesso formato e contenere lo stesso set di attributi relativi al Titolare. Ogni Attestato DEVE contenere dati crittografici diversi per impedire la correlabilità tra gli Attestati Elettronici.

Nota

Standard or Batch Credential Issuance:

L'Utente può configurare la Wallet Solution per emettere Attestati Elettronici in modalità batch o standard e definire la dimensione preferita del batch.

L'intero Issuance Flow può essere suddiviso in due sotto-flussi:

- **Flusso di Richiesta dell'Utente**, che descrive le modalità attraverso le quali l'Utente può richiedere l'Attestato Elettronico. Può essere:
  1. Su iniziativa dell'Utente (**Wallet Initiated**).
  2. Su proposta del Credential Issuer (**Issuer Initiated**).

- **Flusso di Emissione**, che descrive le interazioni tra l'Istanza del Wallet e il Credential Issuer.

Il seguente diagramma mostra il *flusso di richiesta dell'Utente*.

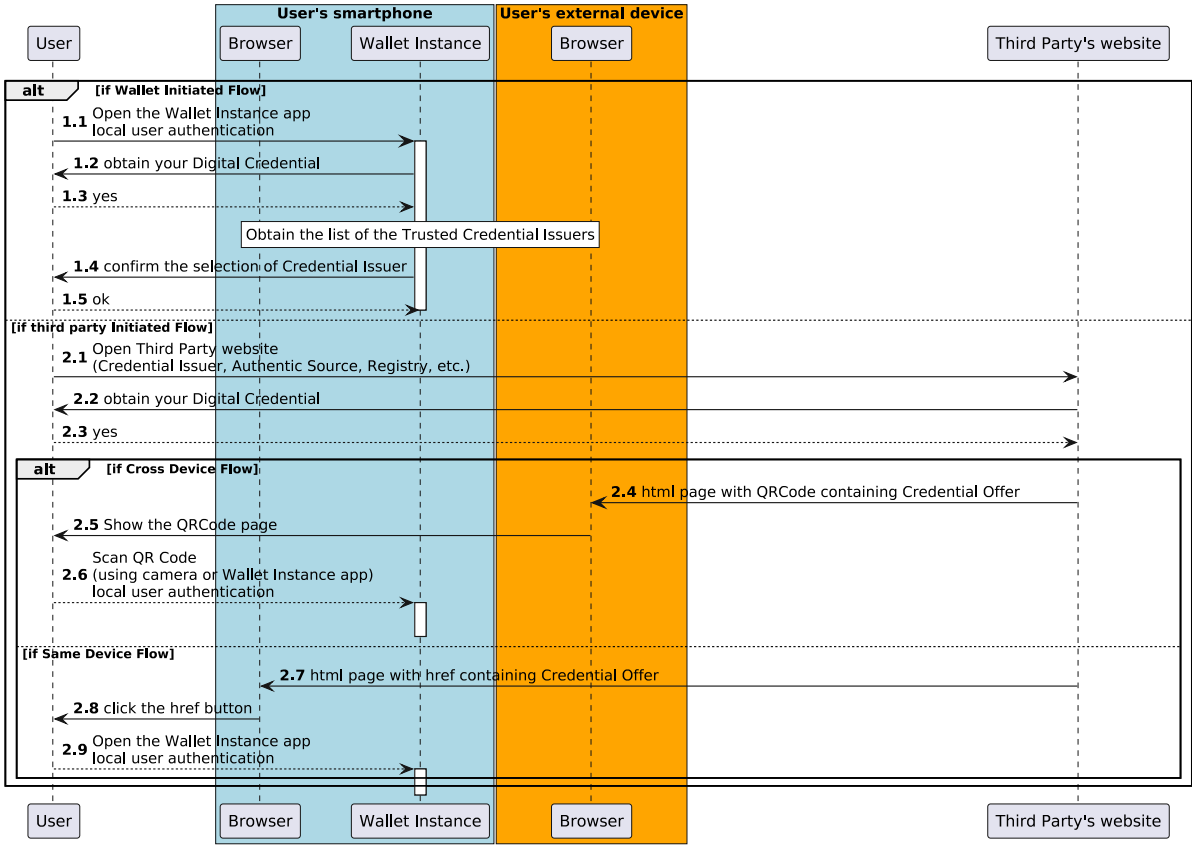


Fig. 12.4: Richiesta dell'Attestato Elettronico da parte dell'Utente - Flusso dettagliato.

Passi 1.1-1.4 (Flusso Avviato dal Wallet): L'Utente, utilizzando l'Istanza del Wallet, seleziona il Credential Issuer tra quelli elencati nella lista delle entità affidabili.

Passi 2.1-2.3 (Flusso Avviato da una Terza Parte): L'Utente, mentre naviga sul sito web del Credential Issuer o di una Terza Parte che supporta il meccanismo di Credential Offer (per esempio una Fonte Autentica), trova un link per ottenere un Attestato Elettronico.

Passi 2.4-2.6 (Cross-Device): La Credential Offer viene presentata come un codice QR. L'Utente lo scansiona con la fotocamera del dispositivo o con lo scanner integrato dell'Istanza del Wallet, attivando l'Istanza del Wallet per recuperare i parametri definiti nella *Tabella dei parametri della Credential Offer* (pagina 478) (*WP_047-048* (pagina 790)).

Passi 2.7-2.10 (Same-Device): La Credential Offer viene presentata come un pulsante href contenente l'URL che consente all'Utente di invocare l'Istanza del Wallet utilizzando il Credential Offer Endpoint.

La Credential Offer può essere inviata per valore (utilizzando il parametro `credential_offer`) o per riferimento (utilizzando il parametro `credential_offer_uri`) come definito nella Sezione 4

di [OpenID4VCI]. Ulteriori dettagli ed esempi non normativi sono forniti nella Sezione *Flusso Credential Offer* (pagina 477).

Il seguente diagramma mostra il *flusso di emissione*.



Fig. 12.5: Emissione degli Attestati Elettronici - Flusso dettagliato.

Una volta completato il *flusso di richiesta dell'Utente*, l'Istanza del Wallet elabora i Metadata del Credential Issuer come definito nella Sezione *Meccanismo di Trust Evaluation* (pagina 140). Inoltre, in caso di emissione di Credenziali in batch, l'Istanza del Wallet DEVE verificare che venga supportata l'emissione in batch tramite l'oggetto `batch_credential_issuance` presente nei metadati del Credential Issuer, da cui l'Istanza del Wallet può ottenere il valore `batch_size`. L'emissione in batch si applica solo ai tipi di Attestato per i quali quei metadati sono dichiarati. Il Credential Issuer DEVE dichiarare se PID e IT-Wallet ID rientrano nel perimetro. Un tipo di Attestato senza tale dichiarazione DEVE essere emesso come singolo Attestato (*CI_203* (pagina 825)).

Nota

Controllo della Federazione: L'Istanza del Wallet deve verificare se il Credential Issuer è membro della Federazione, ottenendo i suoi Metadata specifici per il protocollo (*WP_046* (pagina 790)). Un esempio non normativo di una risposta dall'Endpoint **.well-known/openid-federation** con la **Entity Configuration** e i **Metadata** del Credential Issuer è rappresentato nella sezione credential-issuer-entity-configuration:Entity Configuration del Fornitore di Attestati Elettronici.

Nel caso del flusso avviato dall'Issuer, oltre al controllo della federazione definito sopra, l'Istanza del Wallet DEVE eseguire i seguenti controlli sui parametri della Credential Offer:

- Per ogni identificativo di Attestato Elettronico contenuto nell'array `credential_configuration_ids`, verificare se è supportato dal Credential Issuer (*WP_050* (pagina 790)).
- L'identificativo dell'Authorization Server (se presente) è contenuto nel parametro `authorization_servers` dei Metadata del Credential Issuer (*WP_049* (pagina 790)).

Passi 1-2 ('PAR Request'): L'Istanza del Wallet:

- Crea un nuovo *PKCE code verifier*, una prova di possesso dell'Attestato di Unità di Wallet e un parametro `state` per la *Pushed Authorization Request* (*WP_052* (pagina 790)).
- Fornisce al PAR Endpoint del Credential Issuer i parametri precedentemente elencati, utilizzando il parametro `request` (di seguito *Request Object*) secondo la Sezione 3 di **RFC 9126** per prevenire l'attacco di scambio del *Request URI* (*WP_052* (pagina 790)). La Pushed Authorization Request consente l'autenticazione del client prima di qualsiasi interazione dell'Utente. Questo passaggio permette il rifiuto anticipato di richieste illegittime, prevenendo efficacemente attacchi di spoofing, manomissione e uso improprio delle richieste di autorizzazione.
- DEVE creare il `code_verifier` con una stringa casuale con sufficiente entropia utilizzando i caratteri non riservati con una lunghezza minima di 43 caratteri e una lunghezza massima di 128 caratteri, rendendo impraticabile per un attaccante indovinarne il valore. Il valore DEVE essere generato seguendo la raccomandazione nella Sezione 4.1 di **RFC 7636** (*WP_052a* (pagina 790)).
- Firma questa richiesta utilizzando la chiave privata creata durante la fase di configurazione per ottenere l'Attestato di Unità di Wallet. La relativa chiave pubblica attestata dal Fornitore di Wallet viene fornita all'interno del claim `cnf.jwk` dell'Attestato di Unità di Wallet (*WP_052c* (pagina 790)).
- DEVE utilizzare i parametri `OAuth-Client-Attestation` e `OAuth-Client-Attestation-PoP` secondo OAuth 2.0 Attestation-based Client Authentication [OAUTH-ATTESTATION-CLIENT-AUTH], poiché in questo flusso il Pushed Authorization Endpoint è un endpoint

protetto (*WP_052b* (pagina 790)).

- DEVE specificare i tipi di Credenziali richieste utilizzando sia il parametro `scope`, come richiesto da [OPENID4VC-HAIP], sia il parametro `authorization_details` [RAR **RFC 9396**] (*WP_052d* (pagina 790)).

Nota

JAR [**RFC 9101**] è obbligatorio in questa specifica tecnica per garantire l'integrità end-to-end della richiesta di autorizzazione e di tutti i parametri inclusi nel Request Object. Tuttavia, la Wallet Instance può interagire con Credential Issuer cross-border i cui Authorization Server non seguono questa specifica e potrebbero non supportare l'implementazione di JAR. Per gestire questo scenario, la Wallet Instance DOVREBBE verificare il parametro *require_signed_request_object* nei metadata dell'Authorization Server e decidere in base ad esso se inviare i parametri nel signed Request Object o meno. Per interoperabilità, la Wallet Instance PUÒ duplicare gli stessi parametri nel corpo della richiesta. La Sezione 10.7 di **RFC 9101** fornisce i requisiti di sicurezza su come gestire correttamente questa duplicazione.

Nota

La selezione del metodo di autenticazione dell'Utente, incluso l'uso di `it_l2+document_proof`, è definita in credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente. Il relativo flusso di protocollo è definito in *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482).

Il Credential Issuer esegue i seguenti controlli alla ricezione della *PAR Request*:

1. DEVE validare la firma del *Request Object* utilizzando l'algoritmo specificato nel parametro `alg` dell'header (**RFC 9126**, **RFC 9101**) e la chiave pubblica recuperata dall'Attestato di Unità di Wallet (`cnf.jwk`) referenziato nel *Request Object*, utilizzando il parametro `kid` dell'header JWT.
2. DEVE verificare che l'algoritmo utilizzato per firmare la richiesta nell'header `alg` sia uno di quelli elencati nella Sezione *Algoritmi Crittografici* (pagina 647).
3. DEVE verificare che il `client_id` nel body della *PAR Request* corrisponda al claim `client_id` incluso nel *Request Object*.
4. DEVE verificare che il claim `iss` nel *Request Object* corrisponda al claim `client_id` nel *Request Object* (**RFC 9126**, **RFC 9101**).
5. DEVE verificare che il claim `aud` nel *Request Object* sia uguale all'identificativo del Credential Issuer (**RFC 9126**, **RFC 9101**).

6. DEVE rifiutare la *PAR Request*, se contiene il parametro `request_uri` (**RFC 9126**).
7. DEVE verificare che il *Request Object* contenga tutti i parametri obbligatori i cui valori sono validati secondo la *Tabella dei parametri HTTP* (pagina 596) [derivata da **RFC 9126**].
8. DEVE verificare che il *Request Object* non sia scaduto, controllando il claim `exp`.
9. DEVE verificare che il *Request Object* sia stato emesso in un momento precedente al valore esposto nel claim `iat`. DOVREBBE rifiutare la richiesta se il claim `iat` è lontano dall'ora corrente (**RFC 9126**) di più di *5* minuti.
10. DEVE verificare che il claim `jti` nel *Request Object* non sia stato utilizzato in precedenza dall'Istanza del Wallet identificata dal `client_id`. Ciò consente al Credential Issuer di mitigare gli attacchi di replay (**RFC 7519**).
11. DEVE validare il parametro `OAuth-Client-Attestation-PoP` in base alla Sezione 5 di [OAUTH-ATTESTATION-CLIENT-AUTH].
12. DEVE verificare, se presente, la validità dell'`issuer_state` e la coerenza con il `credential_configuration_id` richiesto nel Request Object.

Di seguito un esempio non normativo di *PAR Request*.

```
POST /as/par HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sIndhbGxldF9uYW1lIjoiV2Fsb
GV0X3YxIiwid2FsbGV0X2xpbmsiOiJodHRwczovL2V4YW1wbGUuY29tL3dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2Fs
bGV0X3ZlcnNpb24iOiIxLjAuMCIsImNsaWVudF9zdGF0dXMiOnsic3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTM
zNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3RhdHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH
0sImV4cCI6MTc0MDE1ODE2N30.j3nx-adTbi3_O0Seddi_-Fqr37MJpDHCI5leCie7iTXApqn_4kOlrDgpmY35pGfZhlB_1
lZUBTnKywNn5LjNow
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA

client_id=$thumprint-of-the-jwk-in-the-cnf-wallet-instance-attestation$&
request=$SIGNED-JWT
```

Di seguito un esempio non normativo dell'header e del body della prova di possesso dell'Attestato di Unità di Wallet (WIA-PoP):

```
{
    "typ": "oauth-client-attestation-pop+jwt",
    "alg": "ES256"
}
```

```
{
    "iss": "47b982369791d08003a7283f059cb0d1",
    "aud": "https://eaa-provider.example.org",
    "iat": 1715842560,
    "jti": "f8555ceb-c65c-4025-9378-b6672b6149af"
}
```

Di seguito un esempio non normativo del *Request Object* firmato senza codifica e firma applicata:

```
{
    "typ": "oauth-authz-req+jwt",
    "alg": "ES256",
    "kid": "b01b8208d9e6cc834d87dc356ab50170"
}
```

```
{
    "jti": "f8555ceb-c65c-4025-9378-b6672b6149af",
    "aud": "https://eaa-provider.example.org",
    "iat": 1715842560,
    "exp": 1715842860,
    "response_type": "code",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "iss": "47b982369791d08003a7283f059cb0d1",
    "state": "fyZiOL9Lf2CeKuNT2JzxiLRDink0uPcd",
    "code_challenge": "E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM",
    "code_challenge_method": "S256",
    "scope": "Education_degree",
    "authorization_details": [
        {
            "type": "openid_credential",
            "credential_configuration_id": "dc_sd_jwt_Education_degree"
        }
    ],
    "redirect_uri": "https://client.example.com/cb"
}
```

Nota

Controllo della Federazione: Il Credential Issuer DEVE verificare che il Fornitore di Wallet faccia parte della federazione.

Nota

Il Credential Issuer DEVE validare la firma dell'Attestato di Unità di Wallet e che non sia scaduto.

Passo 3 ('PAR Response'): Il Credential Issuer fornisce un valore `request_uri` monouso. Il valore `request_uri` emesso DEVE essere vincolato all'identificativo del client (`client_id`) che è stato fornito nel *Request Object*.

Nota

L'entropia del `request_uri` DEVE essere sufficientemente grande. L'adeguata brevità della validità e l'entropia del `request_uri` dipendono dal calcolo del rischio basato sul valore della risorsa protetta. Il tempo di validità DOVREBBE essere inferiore a un minuto e il `request_uri` DEVE includere un valore casuale crittografico di 128 bit o più (**RFC 9101**). L'intero `request_uri` NON DOVREBBE superare i 512 caratteri ASCII per i seguenti due motivi principali (**RFC 9101**):

1. Molti telefoni sul mercato ancora non accettano payload di grandi dimensioni. La restrizione è tipicamente di 512 o 1024 caratteri ASCII.
2. Su una connessione lenta come una connessione mobile 2G, un URL grande causerebbe una risposta lenta; pertanto, l'uso di tale URL non è consigliabile dal punto di vista dell'esperienza utente.

Il Credential Issuer restituisce il `request_uri` emesso all'Istanza del Wallet. Un esempio non normativo della risposta è mostrato di seguito.

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json

{
    "request_uri": "urn:ietf:params:oauth:request_uri:bwc4JK-ESC0w8acc191e-Y1LTC2",
    "expires_in": 60
}
```

Passi 4-5 ('Authorization Request'): L'Istanza del Wallet invia una richiesta di autorizzazione all'Authorization Endpoint del Credential Issuer (*WP_053* (pagina 790)). Poiché parti del contenuto di questa *Authorization Request*, ad esempio il valore del parametro `code_challenge`, sono unici per una particolare *Authorization Request*, l'Istanza del Wallet DEVE utilizzare un valore `request_uri` una sola volta (**RFC 9126**) come da (*WP_053a* (pagina 790)). Il Credential Issuer esegue i seguenti controlli alla ricezione della *Authorization Request*:

1. DEVE trattare i valori `request_uri` come monouso e DEVE rifiutare una richiesta scaduta. Tuttavia, PUÒ consentire richieste duplicate causate da un Utente che ricarica/aggiorna il proprio user-agent (derivato da **RFC 9126**).
2. DEVE identificare la richiesta come risultato del PAR inviato (derivato da **RFC 9126**).
3. DEVE rifiutare tutte le *Authorization Request* che non contengono il parametro `request_uri`, poiché il PAR è l'unico modo per passare la *Authorization Request* dall'Istanza del Wallet (derivato da **RFC 9126**).

```
GET /authorize?client_id=$thumprint-of-the-jwk-in-the-cnf-wallet-instance-attestation$&request_
uri=urn%3Aietf%3Aparams%3Aoauth%3Arequest_uri%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: eaa-provider.example.org
```

Nota

Autenticazione dell'Utente e Consenso: Il Credential Issuer esegue l'autenticazione dell'Utente all'*Authorization Endpoint* secondo credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente. Durante questo passaggio, i Credential Issuer POSSONO chiedere i dettagli di contatto dell'Utente (ad esempio, il loro indirizzo email) per inviare notifiche sugli Attestati Elettronici emessi.

Passi 6-7 ('Authorization Response'): Il Credential Issuer invia un `code` di autorizzazione insieme ai parametri `state` e `iss` all'Istanza del Wallet. L'Istanza del Wallet esegue i seguenti controlli sulla *Authorization Response*:

1. DEVE verificare che la *Authorization Response* contenga tutti i parametri definiti secondo la *Tabella dei parametri della Risposta HTTP* (pagina 600) (*WP_054* (pagina 790)).
2. DEVE verificare che il valore restituito dal Credential Issuer per il parametro `state` sia uguale al valore inviato dall'Istanza del Wallet nel *Request Object* (**RFC 6749**) come specificato in *WP_054a* (pagina 790).
3. DEVE verificare che l'URL del Credential Issuer nel parametro `iss` sia uguale all'identificativo URL previsto del Credential Issuer con cui l'Istanza del Wallet ha iniziato la comunicazione (**RFC 9027**) come specificato in *WP_054b* (pagina 790).

Nota

L'URI di reindirizzamento dell'Istanza del Wallet è un *universal link* o *app link* registrato con il sistema operativo locale, quindi quest'ultimo lo risolverà e passerà la risposta all'Istanza del Wallet.

```
HTTP/1.1 302 Found
Location: https://start.wallet.example.org?code=SplxlOBeZQQYbYS6WxSbIA&state=fyZiOL9Lf2CeKuNT2J
zxiLRDink0uPcd&iss=https%3A%2F%2Feaa-provider.example.org
```

Passi 8-9 ('DPoP Proof' per il Token Endpoint): L'Istanza del Wallet DEVE creare una nuova coppia di chiavi e un nuovo JWT di *DPoP proof* seguendo le istruzioni fornite nella Sezione 4 di (**RFC 9449**) per la richiesta di token al Credential Issuer (*WP_055b* (pagina 790)). Il JWT di *DPoP proof* è firmato utilizzando la chiave privata per DPoP creata dall'Istanza del Wallet per questo scopo (*WP_055c* (pagina 790)). DPoP associa l'Access Token, e opzionalmente il Refresh Token, a una determinata Istanza del Wallet (**RFC 9449**) e mitiga l'uso improprio di token persi o rubati al Credential Endpoint.

Passo 10 ('Token Request'): L'Istanza del Wallet invia una richiesta di token al Token Endpoint del Credential Issuer con un JWT di *DPoP proof* e i parametri: `code`, `code_verifier` e OAuth 2.0 Attestation based Client Authentication (`OAuth-Client-Attestation` e `OAuth-Client-Attestation-PoP`) come specificato in *WP_055* (pagina 790).

L'`OAuth-Client-Attestation` è firmato utilizzando la chiave privata associata all'Istanza del Wallet (*WP_055a* (pagina 790)). La relativa chiave pubblica attestata dal Fornitore di Wallet è fornita all'interno dell'Attestato di Unità di Wallet (claim `cnf.jwk`) come specificato in *WP_055d* (pagina 790). Il Credential Issuer esegue i seguenti controlli sulla *Token Request*:

1. DEVE assicurarsi che il `code` di autorizzazione sia emesso per l'Istanza del Wallet autenticata (**RFC 6749**) e non sia stato replicato.
2. DEVE assicurarsi che il `code` di autorizzazione sia valido e non sia stato utilizzato in precedenza (**RFC 6749**).
3. DEVE assicurarsi che il `redirect_uri` corrisponda al valore incluso nel precedente *Request Object* (vedi Sezione 3.1.3.1. di [OIDC]).
4. DEVE validare il JWT di *DPoP proof*, secondo la Sezione 4.3 di (**RFC 9449**).
5. DEVE verificare il parametro `code_verifier` secondo il meccanismo PKCE, assicurandosi che corrisponda al `code_challenge` associato al codice di autorizzazione come definito nella Sezione 4.6 di **RFC 7636**; in caso contrario, la richiesta DEVE essere rifiutata (*CI_061a* (pagina 825)).

```
POST /token HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwieCI6IjR2dDhNdEFISmlsM
zBDNnpUTmt2c0VVcnlHTEUtQW5BNkc5LV8xa3l5Rk0iLCJ5IjoiTWdiNTFfbjNSRjNtbHNtS3dMd0xtRUFqVmlJM3Q1bTVW
NTI2MFA5MzR3RSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIjoiR0VUIiwiaHR1Ij
oiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0ZWRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.3Tp1Z
lZ05PQYeZUHhiZwaQ1etqnwYwoiJHFR_JHb32381lMJL-8o2rE3VZ8X3yuqrGFfCVeP90Ln4J5r8ASIBg
```

```
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sImNsaWVudF9zdGF0dXMiOnsic
3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTMzNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3Rh
dHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH0sIndhbGxldF9uYW1lIjoiV2FsbGV0X3YxIiwid2FsbGV0X2xpbms
iOiJodHRwczovL2V4YW1wbGUuY29tL3dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2FsbGV0X3ZlcnNpb24iOiIxLjAuMC
IsImV4cCI6MTc0MDE1ODE2N30.j2M6GTp3L7SzDHO6KUDo7PCuqCzO0TsHDYwOezsrAuDres4F8hAhK2SymL6nTMHuInaAQ
M74QlMvfBD2bY9E-w
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
&redirect_uri=https://start.wallet.example.org/cb
```

Passo 11 (‘Token Response‘): Il Credential Issuer valida la richiesta. In caso di successo, l’Issuer fornisce all’Istanza del Wallet un Access Token e, opzionalmente, un Refresh Token, entrambi associati alla chiave DPoP.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "access_token": "eyJ0eXAiOiJhdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwYT
UwMDk2YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsInN1YiI6ImQ0Z
TBiYjM4N2FhMjU1NmZmMzA2OTI1ZmRmYjlhNzY1IiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV0LmlwenMu
aXQvY3JlZGVudGlhbCIsImlhdCI6MTczOTk1Mjk0OCwiZXhwIjoxNzM5OTUzNTQ4LCJqdGkiOiJmOTY1NWNlYi1jNjVjLTQ
wMjUtOTM3OC1iNjY3MmI2MTQ5YmciLCJjbGllbnRfaWQiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsIm
NuZiI6eyJqa3QiOiI5NTE1NzRhZWUxYmI3OTA3YWUxZWMzMTA5ZGIyYjIyNSJ9fQ.XQ3TAo69f2iPNJuMYAS5HdOSN8G5wm
IRMcSR6ocLEHdDgTUNI0TrMk-KapG4fvu5_bdZkjWBJvbjXeuP-Jngpg",
    "refresh_token": "eyJ0eXAiOiJydCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwY
TUwMDk2YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsImNsaWVudF9p
ZCI6IjQ3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV
0LmlwenMuaXQiLCJpYXQiOjE3Mzk5NTI5NDgsIm5iZiI6MTczOTk1MzU0OCwiZXhwIjoxNzQyMzcyNzQ4LCJhdGgiOiJmVU
h5TzJyMlozRFo1M0VzTnJXQmIweFdYb2FOeTU5SWlLQ0Fxa3NtUUVvIiwianRpIjoiYzY5NTVjZWItYzY1Zi00MDI1LTkzN
zgtYjY2NzJiNjE0NWNmIiwiY25mIjp7ImprdCI6Ijk1MTU3NGFlZTFiYjc5MDdhZTFlYzMxMDlkYjJiMjI1In19.qiGM6E-
7zci2-3Nnk4OMD7Tv_leUcRPsFsqaBHDHxEEzsGXLNh9qDbLIBk9sujZGVT9xs-28jZhwD6VT-MGTGw",
    "token_type": "DPoP",
    "expires_in": 3600,
    "authorization_details": [
```

```
        {
            "type": "openid_credential",
            "credential_configuration_id": "dc_sd_jwt_Education_degree",
            "credential_identifiers": [ "education_degree_unibo_2017_l31_informatica", "educati
on_degree_unibo_2019_lm18_informatica", "education_degree_unibo_2022_phd_informatica" ]
        }
    ]
}
```


Un esempio non normativo dell'Access Token DPoP è fornito di seguito.


```
{
    "typ": "at+jwt",
    "alg": "ES256",
    "kid": "c950c0e6fdeb5de50a50096b247af03c"
}
```

```
{
    "iss": "https://eaa-provider.example.org",
    "sub": "d4e0bb387aa2556ff306925fdfb9a765",
    "aud": "https://eaa-provider.example.org",
    "iat": 1739952948,
    "exp": 1739953548,
    "jti": "f9655ceb-c65c-4025-9378-b6672b6149bg",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "cnf": {
        "jkt": "951574aee1bb7907ae1ec3109db2b225"
    }
}
```


Un esempio non normativo del Refresh Token DPoP è fornito di seguito.


```
{
    "typ": "rt+jwt",
    "alg": "ES256",
    "kid": "c950c0e6fdeb5de50a50096b247af03c"
}
```

```
{
    "iss": "https://eaa-provider.example.org",
    "sub": "d4e0bb387aa2556ff306925fdfb9a765",
    "aud": "https://eaa-provider.example.org",
    "iat": 1739952948,
    "exp": 1742372748,
    "nbf": 1739953548,
    "ath": "fUHyO2r2Z3DZ53EsNrWBb0xWXoaNy59IiKCAqksmQEo",
    "jti": "c6955ceb-c65f-4025-9378-b6672b6145cf",
    "client_id": "47b982369791d08003a7283f059cb0d1",
    "cnf": {
```

```
        "jkt": "951574aee1bb7907ae1ec3109db2b225"
    }
}
```

Passo 12 ('Nonce Request'): Secondo la Sezione 7.1 di [OpenID4VCI], l'Istanza del Wallet invia una richiesta HTTP POST al Nonce Endpoint per ottenere un nuovo `c_nonce` che può essere utilizzato per creare la prova di possesso del materiale crittografico per la successiva richiesta al Credential Endpoint (*WP_056a* (pagina 790)).

Di seguito è riportato un esempio non normativo di una *Nonce Request*:

```
POST /nonce HTTP/1.1
Host: eaa-provider.example.org
Content-Length: 0
```

Passo 13 ('Nonce Response'): Il Credential Issuer fornisce il `c_nonce` all'Istanza del Wallet. Il parametro `c_nonce` è una stringa, che DEVE essere imprevedibile e viene utilizzata successivamente dall'Istanza del Wallet nel Passo 16 per creare la(e) prova(e) di possesso della chiave (claim `proofs`) ed è la principale contromisura contro l'attacco di replay della prova della chiave. Si noti che il valore `c_nonce` ricevuto può essere utilizzato per creare la(e) prova(e) finché l'Issuer fornisce all'Istanza del Wallet un nuovo valore `c_nonce`.

Di seguito è riportato un esempio non normativo di una *Nonce Response*:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "c_nonce": "ts_EtUQs0ieiIS1NYNBHEQSoy3ct4gpy-4FZKwHilkY"
}
```

Passo 14 (DPoP Proof per il Credential Endpoint): L'Istanza del Wallet crea un JWT DPoP Proof utilizzando la stessa chiave del **Passo 8** e secondo la Sezione 4 di (**RFC 9449**) (*WP_056b* (pagina 790)).

Nota

Se l'Istanza del Wallet richiede l'emissione in batch di Attestati Elettronici, il flusso prosegue con il **Passo 17**. In caso contrario, vengono eseguiti i passi **15-16**.

Passo 15 (Prova di Possesso della Credenziale): L'Istanza del Wallet per la richiesta delle Credenziali Digitali crea una prova di possesso con il `c_nonce` ottenuto nel **Passo 13** utilizzando la chiave privata utilizzata per il DPoP. Il valore del `jwk` nel parametro proof DEVE essere uguale alla chiave pubblica referenziata nel DPoP (*WP_056c* (pagina 790)).

Passo 16 ('Credential Request'): L'Istanza del Wallet invia una richiesta per l'Attestato Elettronico al Credential Endpoint. Questa richiesta DEVE includere l'Access Token, il JWT

di *DPoP proof*, il tipo di Attestato Elettronico, la prova (che dimostra il possesso del materiale crittografico) come specificato in *WP_056* (pagina 790). Il parametro `proofs` DEVE essere un oggetto che contiene la prova di possesso del materiale crittografico a cui sarà associato l'Attestato Elettronico emesso. Per verificare la prova, il Credential Issuer conduce i seguenti controlli al Credential Endpoint:

1. La prova JWT DEVE includere tutti i claim richiesti come specificato nella tabella della Sezione credential-issuance-endpoint:Token Request.
2. La prova della chiave DEVE essere esplicitamente tipizzata utilizzando i parametri dell'header come definito per il rispettivo tipo di prova.
3. Il parametro dell'header `alg` DEVE indicare un algoritmo di firma digitale asimmetrica registrato e NON DEVE essere impostato su *none*.
4. La firma sulla prova della chiave DEVE essere verificata utilizzando la chiave pubblica specificata nel parametro dell'header.
5. Il parametro dell'header NON DEVE contenere una chiave privata.
6. La firma sul JWT di Key Attestation, come valore del parametro di intestazione `key_attestation`, DEVE essere verificata utilizzando la chiave pubblica del Wallet Provider, identificata dal parametro di intestazione kid all'interno del JWT di Key Attestation.
7. Se un valore `c_nonce` è stato precedentemente fornito dal server, il claim `nonce` nel JWT DEVE corrispondere a questo valore `c_nonce`. Inoltre, l'istante di creazione del JWT, come indicato dal claim `iat` o da un timestamp gestito dal server tramite il claim `nonce`, DEVE essere all'interno di una finestra temporale accettabile come determinato dal server.

Nota

Il Credential Issuer DEVE registrare tutti gli Attestati Elettronici emessi per la loro successiva revoca, se necessario.

Nota

È RACCOMANDATO che la chiave pubblica contenuta nel `jwt_proof` sia generata specificamente per l'Attestato Elettronico richiesto (nuova chiave crittografica) per garantire che diversi Attestati Elettronici emessi non condividano la stessa chiave pubblica, rimanendo così non collegabili tra loro.

Un esempio non normativo della *Credential Request* è fornito di seguito.

```
POST /credential HTTP/1.1
  Host: eaa-provider.example.org
```

```
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA
```

```
{
    "credential_identifier": "education_degree_unibo_2017_l31_informatica",
    "proofs": {
        "jwt": [
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESGdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TlRRME1U
Y3lNVEkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkZlEuWndqZUpUQTNnc3Z0Mk1NdzlTMy1iZzREblY1TGRRRXB6OUYxcmVCRWlDblRuaURVa
E9jS010RzVnNmZwRFBVMWRQMjBzdTlXVXR1Rld6c0d1b0dhNFEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.2Z
QF2kYONuzkqIJ7nJxSl41-f0bTVSBdDpLFSrhA6PfmrcQJzBlTsv7FuGtXN-6mF3tZroklA2zjLmcaSCP3QQ"
        ]
    }
}
```

Dove un esempio non normativo del contenuto decodificato del parametro `jwt` è rappresentato di seguito, senza codifica e firma.

```
{
    "typ": "openid4vci-proof+jwt",
    "alg": "ES256",
    "jwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "4Fi3wDIgfbeQK624NyR7XOvAk3shaWy935yv5SHBqMw",
        "y": "D-R10rPxM0yqTJF6I3v-73USCU_6FHazyIJPLEsyUFE"
```

```
    },
    "key_attestation": "eyJhbGciOiJFUzI1NiIsImtpZCI6IjV0NVlZcEJoTi1FZ0lFRUk1aVV6cjZyME1SMDJMblZ
RME9tZWttTktjalkiLCJ0cnVzdF9jaGFpbiI6WyJleUpoYkdjaU9pSkZVei4uLjZTMEEiLCJleUpoYkdjaU9pSkZVei4uLm
pKTEEiLCJleUpoYkdjaU9pSkZVei4uLkg5Z3ciXSwidHlwIjoia2V5YXR0ZXN0YXRpb24rand0In0.eyJhdHRlc3RlZF9rZ
XlzIjpbeyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6IjRGaTN3RElnZmJlUUs2MjROeVI3WE92QWszc2hhV3k5MzV5
djVTSEJxTXciLCJ5IjoiRC1SMTByUHhNMHlxVEpGNkkzdi03M1VTQ1VfNkZIYXp5SUpQTEVzeVVGRSJ9XSwia2V5X3N0b3J
hZ2Vfc3RhdHVzIjp7InN0YXR1cyI6eyJzdGF0dXNfbGlzdCI6eyJpZHgiOjQxMiwidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW
9uX3VybC9zdGF0dXNaXN0cy8xIn19LCJleHAiOjE1NDQxNzIxMjR9LCJpYXQiOjE2ODcyODExOTUsImV4cCI6MTY4NzI4O
DM5NSwia2V5X3N0b3JhZ2UiOlsiaXNvXzE4MDQ1X21vZGVyYXRlIl0sInVzZXJfYXV0aGVudGljYXRpb24iOlsiaXNvXzE4
MDQ1X21vZGVyYXRlIl19.LG9hkaoGmznUAL43gytXyEOoH7J-kKZQj_sG7Row2DUl0OYArV3mJnDmIuK_4-YVPg4AM03xV0
28BvFRwv5XDw"
}

{
    "iss": "47b982369791d08003a7283f059cb0d1",
    "aud": "https://eaa-provider.example.org",
    "iat": 1705570055,
    "nonce": "ts_EtUQs0ieiIS1NYNBHEQSoy3ct4gpy-4FZKwHilkY"
}
```


Passo 17 (Generazione di nuove chiavi per le Credenziali Digitali): L'Istanza del Wallet genera N nuove coppie di chiavi per le Credenziali, dove il numero di coppie di chiavi (N) è determinato dal valore definito in `batch_size` (*WP_058a* (pagina 790)).

Passo 18 (Prove di possesso delle credenziali): L'istanza del Wallet DEVE generare N *key proofs* utilizzando il `c_nonce` fornito nel **Passo 13** e una per ogni Credenziale presente nel batch. Il numero di *key proofs* (N) è definito dal valore `batch_size` (*WP_058b* (pagina 790)).

Nota

Il valore `c_nonce` in tutte le proof jwt è identico e non è necessario ottenere valori del nonce diversi.

Passo 19 (Batch Credential Request): L'Istanza del Wallet invia una richiesta per il batch di Credenziali Digitali al Credential Endpoint. Questa richiesta DEVE includere l'Access Token, il JWT DPoP Proof, il tipo di Credenziale e le proofs (che dimostrano il possesso delle chiavi). Il parametro `proofs` DEVE essere impostato utilizzando un oggetto JSON contenente due o più prove di possesso dei materiali crittografici a cui sarà associato il batch di Credenziali Digitali emesso. Per verificare le proofs, il Credential Issuer, oltre ai controlli definiti al **Passo 16**, deve assicurarsi che l'attributo `jwk` in ogni proof sia univoco (*WP_058* (pagina 790)).

```
POST /credential HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
```

VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA


```
{
    "credential_identifier": "education_degree_unibo_2017_l31_informatica",
    "proofs": {
        "jwt": [
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2Iiwiandrijp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESWdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TlRRME1U
Y3lNVEkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkZlEuWndqZUpUQTNnc3Z0Mk1NdzlTMy1iZzREblY1TGRRRXB6OUYxcmVCRWlDblRuaURVa
E9jS010RzVnNmZwRFBVMWRQMjBzdTlXVXR1Rld6c0d1b0dhNFEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.2Z
QF2kYONuzkqIJ7nJxSl41-f0bTVSBdDpLFSrhA6PfmrcQJzBlTsv7FuGtXN-6mF3tZroklA2zjLmcaSCP3QQ",
```

```
            "eyJ0eXAiOiJvcGVuaWQ0dmNpLXByb29mK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIi
wiY3J2IjoiUC0yNTYiLCJ4IjoiNEZpM3dESWdmYmVRSzYyNE55UjdYT3ZBazNzaGFXeTkzNXl2NVNIQnFNdyIsInkiOiJEL
VIxMHJQeE0weXFUSkY2STN2LTczVVNDVV82RkhhenlJSlBMRXN5VUZFIn0sImtleV9hdHRlc3RhdGlvbiI6ImV5SmhiR2Np
T2lKRlV6STFOaUlzSW10cFpDSTZJalYwTlZsWmNFSm9UaTFGWjBsRlJVazFhVlY2Y2paeU1FMVNNREpNYmxaUk1FOXRaV3R
0VGt0amFsa2lMQ0owY25WemRGOWphR0ZwYmlJNld5SmxlVXBvWWtkamFVOXBTa1pWZWk0dUxqWlRNRUVpTENKbGVVcG9Za2
RqYVU5cFNrWlZlaTR1TG1wS1RFRWlMQ0psZVVwb1lrZGphVTlwU2taVmVpNHVMa2c1WjNjaVhTd2lkSGx3SWpvaWEyVjVZW
FIwWlhOMFlYUnBiMjRyYW5kMEluMC5leUpoZEhSbGMzUmxaRjlyWlhseklqcGJleUpqY25ZaU9pSlFMVEkxTmlJc0ltdDBl
U0k2SWtWRElpd2llQ0k2SWpSR2FUTjNSRWxuWm1KbFVVczJNalJPZVZJM1dFOTJRV3N6YzJoaFYzazVNelY1ZGpWVFNFSnh
UWGNpTENKNUlqb2lSQzFTTVRCeVVIaE5NSGx4VkVwR05ra3pkaTAzTTFWVFExVmZOa1pJWVhwNVNVcFFURVZ6ZVZWR1JTSj
lYU3dpYTJWNVgzTjBiM0poWjJWZmMzUmhkSFZ6SWpwN0luTjBZWFIxYzE5c2FYTjBJanA3SW1sa2VDSTZOREV5TENKMWNta
2lPaUpvZEhSd2N6b3ZMM0psZG05allYUnBiMjVmZFhKc0wzTjBZWFIxYzJ4cGMzUnpMekVpZlN3aVpYaHdJam94TXpBM09E
QXpOVFkwZlN3aWFXRjBJam94TmpnM01qZ3hNVGsxTENKbGVIQWlPakUyT0RjeU9EZ3pPVFVzSW10bGVWOXpkRzl5WVdkbEl
qcGJJbWx6YjE4eE9EQTBOVjl0YjJSbGNtRjBaU0pkTENKMWMyVnlYMkYxZEdobGJuUnBZMkYwYVc5dUlqcGJJbWx6YjE4eE
9EQTBOVjl0YjJSbGNtRjBaU0pkZlEuUGRFYWhhVlAwT2dqOTRwYTdiSGcxY1BKVmF6bHRibGJORHNxU2R1QVNDUkxjY1Ayc
GxCaDYwNHZNcWp5WHpSV3pDN0xCbXZpQ29IN1hjVEJReXJKaUEifQ.eyJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M
2YwNTljYjBkMSIsImF1ZCI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0IjoxNzA1NTcwMDU1LCJl
eHAiOjE3Nzg5MTQ1NjAsIm5vbmNlIjoidHNfRXRVUXMwaWVpSVMxTllOQkhFUVNveTNjdDRncHktNEZaS3dIaWxrWSJ9.mT
hywjS7lYdmYpR3FY0cVVQJlqaiOPfnEmQU2UcNt4mP6Gl6qgAyhuwqAZoAx5yEeZTzRVzwt44NMrk6mmra8w"
        ]
    }
}
```


Il contenuto decodificato degli elementi `jwt` nell'array `jwt` è simile a quanto spiegato nel **Passo 16**.

Passi 20-24 (Credential Response): Il Credential Issuer DEVE convalidare il *JWT DPoP Proof* in base alle istruzioni definite all'interno della Sezione 4.3 di (**RFC 9449**) e verificare se l'*Access Token* è valido e idoneo per la Credenziale richiesta. Il Credential Issuer DEVE convalidare tutte le prove fornite nel parametro `proofs` (**Passo 15** e **Passo 18**) a cui le nuove Credenziali DEVONO essere vincolate, secondo quanto riportato nell'Appendice F1 di Open ID4VCI. Se tutti i controlli hanno esito positivo, il Credential Issuer restituisce la Credenziale emessa all'interno del parametro `credentials`. Il numero di elementi nell'array `credentials` corrisponde al numero di chiavi fornite dall'Istanza del Wallet tramite il parametro `proofs` (**Passo 16** e **Passo 19**). L'Istanza del Wallet DEVE eseguire i seguenti controlli prima di procedere con l'archiviazione sicura delle Credenziali:

1. DEVE verificare che il PID/(Q)EAA contenuto nella *Credential Response* contenga tutti i parametri obbligatori e i valori siano validati secondo la *Tabella dei parametri della Credential Response* (pagina 616) (*WP_059* (pagina 790)).

2. DEVE verificare l'integrità dell'Attestato Elettronico verificando la firma utilizzando l'algoritmo specificato nel parametro dell'header `alg` di SD-JWT (*Modello di Dati degli Attestati Elettronici* (pagina 379)) e la chiave pubblica che è identificata utilizzando l'header `kid` dell'SD-JWT (*WP_062a* (pagina 790)).

3. DEVE verificare che l'Attestato o gli Attestati Elettronici ricevuti (nel claim `credential`)

corrispondano al tipo di Attestato Elettronico richiesto e sia conforme allo schema specifico di quell'Attestato Elettronico definito in *Modello di Dati degli Attestati Elettronici* (pagina 379) (*WP_060* (pagina 790)).

4. DEVE elaborare e verificare l'Attestato Elettronico nel formato SD-JWT VC (secondo la Sezione 4 di SD-JWT) o nel formato mdoc-CBOR (*WP_062* (pagina 790)).
5. DEVE verificare la Trust Chain nell'header dell'SD-JWT VC per verificare che il Credential Issuer sia affidabile (*WP_061* (pagina 790)).

Se i controlli sopra hanno successo, l'Istanza del Wallet richiede il consenso dell'Utente per memorizzare l'Attestato o gli Attestati Elettronici. Dopo aver ricevuto il consenso, l'Istanza del Wallet memorizza in modo sicuro l'Attestato Elettronico (*WP_063* (pagina 790)).

Di seguito è riportato un esempio non normativo di una risposta di successo contenente un Attestato Elettronico nel formato SD-JWT VC.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
            "credential": "eyJ0eXAiOiJ2YytzZC1qd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRl
YjVkZTUwYTUwMDk2YjI0N2FmMDNjIn0.eyJfc2QiOlsiQ1JJQkdpbWhhbE1TSzhLZUxmNzg2N0w4cHV6MEZnRTVaS1VXLW1
2N0hiYyIsIlhhZjVJZFVGc0UzYWtabEszT0E5d3dHcjJKcVAwUU01M3BBY2hiempRZmMiLCJjR2FuQVdySG9WQVoyalBhNX
k0SzE3U0xpYWFKcGRNUF9PdnBmTGx0VWJjIiwiNEFuZU1ZVVAxRWh3emRHdkRQOEhobnRaRGN1ejZrOHhHWVJ0NXo0SHh0S
SIsIjRuYTVXSHRMYzdrYnNxdHFVaHd6WXdVdUQtY3hKVmdENmRaLTl0ZUdhZ3MiLCJiZS10a2U3YVU0WmhDNWUxcGZqRjcx
UWpFRzRsZG1IaFRoUFl1TnQyOHo0IiwiSnNkaHhTMWRVTTJaN29MUmZYWVJvOFFEaDJ4M1dQNkdILUJnY21DdzJGayIsIlV
obXBRSy1HS3hzaHJwXzZwYVpfZzROVG5fX29aeVdOb01zTGNaMUhlMjgiLCJOdDdQU3RxMkEyWkliUHBHdTJpdmVSek9rbW
pYUEN1V0RBdy0tdktSQTBjIiwiV3Rtck5JVzFTVkN0UnZNUF9UM2YtRGlhRUdHNS1iVk5rWGJRckowRnpzRSJdLCJleHAiO
jE3NDY1MTU5NzIsImlzcyI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmlwenMuaXQiLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25v
V1hGWkFmSGt4WnNSR0M5WHMiLCJzdGF0dXMiOnsic3RhdHVzX2F0dGVzdGF0aW9uIjp7ImNyZWRlbnRpYWxfaGFzaF9hbGc
iOiJzaGEtMjU2In19LCJ2Y3QiOiJFZHVjYXRpb25fZGVncmVlIiwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOnsiandrIj
p7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoicFZVM2phdHU0YTN0azljOWFvd1ZnTHlCQl9ySjdNLTNXbGprMWVqV
XoyRSIsInkiOiJUTDVPTnZSLUlnYXJuZ3J6NWpkdnNwb2ZmekZ3Y2pQUnRGVWtlbmVIRUkwIn19fQ.v9ynFXhKXPOhQSMmu
LvIBKRWfPEPDf4QwDoNmDOjMROxr5J4Hshh9mBEM5qohH_PDE62i1TLc36C65jFYa7x3A~WyJjQ0ZDeXljV1J4alZINkZUR
VR5OTd3IiwidGF4X2lkX2NvZGUiLCJSU1NNUkE4MFIwMUg1MDFCIl0~...~WyIwQUx5SzRfUi1aVUpTekVKdW5HTFdRIiwi
aWF0IiwiMTc0NzExOTU5NSJd"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
}
```

Di seguito è riportato un esempio non normativo di una risposta di successo contenente un Attestato Elettronico nel formato mdoc.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
            "credential": "omppc3N1ZXJBdXRohEOhASahG...ArQwggKwMIICVqADAgEC"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
}
```

Di seguito è riportato un esempio non normativo di una risposta di successo contenente un batch di Attestati Elettronici nel formato SD-JWT VC.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
    "credentials": [
        {
            "credential": "eyJ0eXAiOiJ2YytzZC1qd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRl
YjVkZTUwYTUwMDk2YjI0N2FmMDNjIn0.eyJfc2QiOlsiQ1JJQkdpbWhhbE1TSzhLZUxmNzg2N0w4cHV6MEZnRTVaS1VXLW1
2N0hiYyIsIlhhZjVJZFVGc0UzYWtabEszT0E5d3dHcjJKcVAwUU01M3BBY2hiempRZmMiLCJjR2FuQVdySG9WQVoyalBhNX
k0SzE3U0xpYWFKcGRNUF9PdnBmTGx0VWJjIiwiNEFuZU1ZVVAxRWh3emRHdkRQOEhobnRaRGN1ejZrOHhHWVJ0NXo0SHh0S
SIsIjRuYTVXSHRMYzdrYnNxdHFVaHd6WXdVdUQtY3hKVmdENmRaLTl0ZUdhZ3MiLCJiZS10a2U3YVU0WmhDNWUxcGZqRjcx
UWpFRzRsZG1IaFRoUFl1TnQyOHo0IiwiSnNkaHhTMWRVTTJaN29MUmZYWVJvOFFEaDJ4M1dQNkdILUJnY21DdzJGayIsIlV
obXBRSy1HS3hzaHJwXzZwYVpfZzROVG5fX29aeVdOb01zTGNaMUhlMjgiLCJOdDdQU3RxMkEyWkliUHBHdTJpdmVSek9rbW
pYUEN1V0RBdy0tdktSQTBjIiwiV3Rtck5JVzFTVkN0UnZNUF9UM2YtRGlhRUdHNS1iVk5rWGJRckowRnpzRSJdLCJleHAiO
jE3NDY1MTU5NzIsImlzcyI6Imh0dHBzOi8vZWFhLXByb3ZpZGVyLmlwenMuaXQiLCJzdWIiOiJOemJMc1hoOHVEQ2NkN25v
V1hGWkFmSGt4WnNSR0M5WHMiLCJzdGF0dXMiOnsic3RhdHVzX2F0dGVzdGF0aW9uIjp7ImNyZWRlbnRpYWxfaGFzaF9hbGc
iOiJzaGEtMjU2In19LCJ2Y3QiOiJFZHVjYXRpb25fZGVncmVlIiwiX3NkX2FsZyI6InNoYS0yNTYiLCJjbmYiOnsiandrIj
p7Imt0eSI6IkVDIiwiY3J2IjoiUC0yNTYiLCJ4IjoicFZVM2phdHU0YTN0azljOWFvd1ZnTHlCQl9ySjdNLTNXbGprMWVqV
XoyRSIsInkiOiJUTDVPTnZSLUlnYXJuZ3J6NWpkdnNwb2ZmekZ3Y2pQUnRGVWtlbmVIRUkwIn19fQ.v9ynFXhKXPOhQSMmu
LvIBKRWfPEPDf4QwDoNmDOjMROxr5J4Hshh9mBEM5qohH_PDE62i1TLc36C65jFYa7x3A~WyJjQ0ZDeXljV1J4alZINkZUR
VR5OTd3IiwidGF4X2lkX2NvZGUiLCJSU1NNUkE4MFIwMUg1MDFCIl0~...~WyIwQUx5SzRfUi1aVUpTekVKdW5HTFdRIiwi
aWF0IiwiMTc0NzExOTU5NSJd"
        },

        {
            "credential": "YXNkZnNhZGZkamZqZGFza23....29tZTIzMjMyMzIzMjMy"
        }
    ],
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942"
```

```
}
```

Nota

Quando l'Istanza del Wallet riceve un nuovo batch contentente le stesse credenziali con gli stessi claim, il Wallet DEVE eliminare le Credenziali precedenti (*WP_073a* (pagina 790)).

Nota

Se l'Attestato Elettronico richiesto non può essere emesso immediatamente e richiede più tempo, il Credential Issuer DOVREBBE supportare il Deferred Flow (**passo 27**) come specificato nella Sezione credential-issuance-endpoint:Deferred Endpoint (*WP_065–067* (pagina 790)). Inoltre, in caso di emissione in batch, lo stesso `transaction_id` si riferisce a tutte le Credenziali richieste nel batch.

Passo 25 (Notification Request): Secondo la Sezione 11.1 di [OpenID4VCI], il Wallet invia una richiesta HTTP POST all'Endpoint di notifica utilizzando il tipo di supporto *application/json* come nel seguente esempio non normativo (*WP_064* (pagina 790)).

```
POST /notification HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA

{
    "notification_id": "dab8ef51-fb43-43a5-a5c1-247c93ddb942",
    "event": "credential_accepted"
}
```

Passo 26 ('Notification Response'): Quando il Credential Issuer ha ricevuto con successo la *Notification Request* dal Wallet, DEVE rispondere con un codice di stato HTTP *204* come raccomandato nella Sezione 11.2 di [OpenID4VCI]. Di seguito è riportato un esempio non normativo di risposta a una *Notification Request* riuscita:

```
HTTP/1.1 204 No Content
```

Nota

Nel sistema della Fonte Autentica talvolta sono disponibili molteplici set di dati pertanto l'Utente potrebbe essere interessato a ottenere più di una Credenziale. In questi casi, L'Issuance flow rimane lo stesso descritto in *Flussi Dettagliati per l'Emissione di Attestati Elettronici* (pagina 447). Nella Token Response (**Passo 11**), il Fornitore di Credenziali genera un identificativo univoco (`credential_identifier`) per ciascun set di dati fornito nel parametro `AttributeClaims` del servizio PDND *Get Attribute Claims* (pagina 634). Pertanto, l'array `credential_identifiers` fornito nel parametro `authorization_details` contiene gli identificativi di ciascun set di dati. Successivamente, il Wallet invia una Credential Request (**Passo 16**) per ciascun identificativo, ottenendo più Credenziali distinte che vengono mostrate individualmente all'Utente per la loro accettazione. Infine, il Fornitore di Credenziali viene informato dal Wallet dell'esito tramite il credential-issuance-endpoint:Notification Endpoint (*WP_057* (pagina 790)).

Nota

Per gli Attestati Elettronici emessi in batch, il `notification_id` si riferisce a tutti gli attestati emessi in batch. La Notification Response (ad esempio `credential_accepted` o `credential_stored`) si applica pertanto a tutti gli attestati contenuti nel batch; qualsiasi errore parziale viene trattato come errore dell'intero batch.

Refresh Token Flow

Per utilizzare gli Endpoint Deferred, Credential Request e Notification, l'Istanza del Wallet DEVE presentare un Access Token DPoP valido al Credential Issuer. Tuttavia, quando questi endpoint sono utilizzati nel Deferred Flow, per la riemissione o la notifica dell'eliminazione di un Attestato Elettronico, l'Access Token potrebbe scadere, poiché è progettato per avere una breve durata e queste azioni POSSONO verificarsi giorni dopo (*WP_066b* (pagina 790)). Per affrontare questo problema, la specifica RACCOMANDA l'uso dei Refresh Token.

Un Access Token ottenuto come risultato di un Refresh Token Flow DEVE essere limitato al:

- Deferred Endpoint, per ottenere un nuovo Attestato Elettronico dopo il tempo impostato nel parametro `interval` o quando viene notificato come pronto per essere emesso (*WP_066* (pagina 790));
- Notification Endpoint, per notificare l'eliminazione di un Attestato Elettronico al Credential Issuer;

- Credential Endpoint, per aggiornare un Attestato Elettronico che è già presente nell'Istanza del Wallet (chiamato anche riemissione dell'Attestato Elettronico, vedi sezione *Re-issuance Flow* (pagina 473)).

Per mitigare l'impatto di un Refresh Token rubato, i Refresh Token DEVONO essere DPoP. Questi aspetti sono dettagliati e discussi nella Sezione *Considerazioni di Sicurezza* (pagina 472).

La figura seguente mostra come ottenere un nuovo Access Token DPoP e un nuovo Refresh Token DPoP dal Token Endpoint.

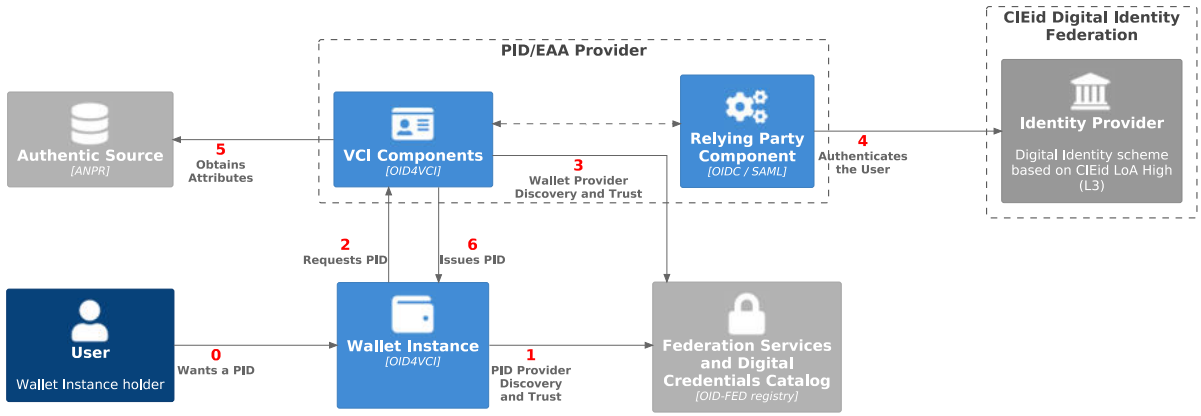


Fig. 12.6: Refresh Token Flow.

Nota

L'aggiornamento di un Token può essere attivato da diverse azioni (ad esempio, l'eliminazione di un Attestato Elettronico da parte dell'Utente). In ogni caso, si suppone che le Istanze del Wallet siano in esecuzione e che il relativo materiale crittografico sia sbloccato.

Passo 1: L'Istanza del Wallet DEVE creare un nuovo JWT di *DPoP proof* e una nuova prova di possesso dell'Attestato di Unità di Wallet per la richiesta di token del Credential Issuer (*WP_068a* (pagina 790)).

Passo 2: Per aggiornare un Access Token vincolato a DPoP, l'Istanza del Wallet invia una richiesta di token utilizzando il parametro `grant_type` impostato su `refresh_token`, includendo l'header DPoP e gli header di OAuth Client Attestation (*WP_068* (pagina 790)). Un esempio non normativo della richiesta di token per un Access Token DPoP utilizzando un Refresh Token è mostrato di seguito.

```
POST /token HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
```

```
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwieCI6IjR2dDhNdEFISmlsM
zBDNnpUTmt2c0VVcnlHTEUtQW5BNkc5LV8xa3l5Rk0iLCJ5IjoiTWdiNTFfbjNSRjNtbHNtS3dMd0xtRUFqVmlJM3Q1bTVW
NTI2MFA5MzR3RSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIjoiR0VUIiwiaHR1Ij
oiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0ZWRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.3Tp1Z
lZ05PQYeZUHhiZwaQ1etqnwYwoiJHFR_JHb32381lMJL-8o2rE3VZ8X3yuqrGFfCVeP90Ln4J5r8ASIBg
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtpZCI6IjBiNDk4ZGRlMDkxNzJhZGE3MDFkMDdlYjZmOTg2N
2FkIiwidHlwIjoib2F1dGgtY2xpZW50LWF0dGVzdGF0aW9uK2p3dCIsIng1YyI6WyJNSUlEcWpDQ0FwS2dBd0lCQWdJRVNM
TkV2REEgLi4uIiwiTUlJQ3d6Q0NBYXNDQ1FDS1Z5OWVLanZpK2pBIC4uLiIsIk1JSURURENDQWpTZ0F3SUJBZ0lKQVBsblF
ZSC4uLiJdfQ.eyJzdWIiOiJodHRwczovL3dhbGxldC1zb2x1dGlvbi5leGFtcGxlLm9yZyIsImNuZiI6eyJqd2siOnsiY3J
2IjoiUC0yNTYiLCJrdHkiOiJFQyIsIngiOiI0SE5wdEkteHIycGp5UkpLR01uejRXbWRuUURfdUpTcTRSOTVOajk4YjQ0Ii
wieSI6IkxJWm5TQjM5dkZKaFlnUzNrN2pYRTRyMy1Db0dGUXdadFBCSVJxcE5scmcifX0sImNsaWVudF9zdGF0dXMiOnsic
3RhdHVzIjp7InN0YXR1c19saXN0Ijp7ImlkeCI6MTMzNywidXJpIjoiaHR0cHM6Ly9yZXZvY2F0aW9uX3VybC93aWEtc3Rh
dHVzbGlzdHMvNDIifX0sImV4cCI6MTMwMzQ5Nzc4MH0sIndhbGxldF9uYW1lIjoiV2FsbGV0X3YxIiwid2FsbGV0X2xpbms
iOiJodHRwczovL2V4YW1wbGUuY29tL3dhbGxldC9kZXRhaWxfaW5mby5odG1sIiwid2FsbGV0X3ZlcnNpb24iOiIxLjAuMC
IsImV4cCI6MTc0MDE1ODE2N30.j2M6GTp3L7SzDHO6KUDo7PCuqCzO0TsHDYwOezsrAuDres4F8hAhK2SymL6nTMHuInaAQ
M74QlMvfBD2bY9E-w
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWNsaWVudC1hdHRlc3RhdGlvbi1wb
3Arand0In0.ew0KICAiaXNzIjogIiA0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsDQogICJhdWQiOiAiaHR0
cHM6Ly9hcy5leGFtcGxlLmNvbSIsDQogICJqdGkiOiAiZDI1ZDAwYWItNTUyYi00NmZjLWFlMTktOThmNDQwZjI1MDY0Iiw
NCiAgImlhdCI6IDE3NDAxNTg2MTcNCn0.B0KOkGi9vMxf3H2Y8rrF-mdLNsuluTvAUbjFfL1Hi-gdaPW7-8ziS9uVh7aTnS
AHKWzMfkZLv5q-bxhkglR4PA
```

```
grant_type=refresh_token
&refresh_token=eyJ0eXAiOiJydCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6ImM5NTBjMGU2ZmRlYjVkZTUwYTUwMDk2
YjI0N2FmMDNjIn0.eyJpc3MiOiJodHRwczovL2VhYS1wcm92aWRlci53YWxsZXQuaXB6cy5pdCIsImNsaWVudF9pZCI6IjQ
3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwiYXVkIjoiaHR0cHM6Ly9lYWEtcHJvdmlkZXIud2FsbGV0Lmlwen
MuaXQiLCJpYXQiOjE3Mzk5NTI5NDgsIm5iZiI6MTczOTk1MzU0OCwiZXhwIjoxNzQyMzcyNzQ4LCJhdGgiOiJmVUh5TzJyM
lozRFo1M0VzTnJXQmIweFdYb2FOeTU5SWlLQ0Fxa3NtUUVvIiwianRpIjoiYzY5NTVjZWItYzY1Zi00MDI1LTkzNzgtYjY2
NzJiNjE0NWNmIiwiY25mIjp7ImprdCI6Ijk1MTU3NGFlZTFiYjc5MDdhZTFlYzMxMDlkYjJiMjI1In19.qiGM6E-7zci2-3
Nnk4OMD7Tv_leUcRPsFsqaBHDHxEEzsGXLNh9qDbLIBk9sujZGVT9xs-28jZhwD6VT-MGTGw
```

Passaggio 3: Il Credential Issuer valida la richiesta sulla base dei seguenti controlli:

- DEVE validare il parametro OAuth-Client-Attestation-PoP in base alla Sezione 5 di [OAUTH-ATTESTATION-CLIENT-AUTH].
- DEVE validare il JWT di *DPoP proof*, secondo la Sezione 4.3 di (RFC 9449).
- DEVE verificare che il Refresh Token non sia scaduto, non sia revocato e sia associato allo stesso set di chiavi DPoP di quelle utilizzate nel JWT di *DPoP proof*.

Se i controlli della richiesta hanno successo, il Credential Issuer genera un nuovo Access Token e un nuovo Refresh Token e questi DEVONO essere entrambi associati alla chiave DPoP. Sia l'Access Token che il Refresh Token vengono quindi inviati all'Istanza del Wallet.

Un esempio non normativo di una risposta di successo è mostrato di seguito.

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
Cache-Control: no-store

{
    "access_token": "eyJ0eXAiOiJhdCtqd3QiLCJhbGciOiJFU..",
    "refresh_token": "eyC3fiLdCtqd3QiLCJhbGciOiCL3..",
    "token_type": "DPoP",
    "expires_in": 3600
}
```

Se il Refresh Token è scaduto o non valido, il Credential Issuer DEVE emettere un errore, utilizzando il claim *error type* impostato su `invalid_grant`. Pertanto, per ottenere l'Attestato Elettronico è necessario un flusso di emissione che autentichi l'Utente, come definito nella Sezione *Issuance Flow* (pagina 447).

Considerazioni di Sicurezza

Per mitigare i rischi di compromissione del Refresh Token, sono richieste le seguenti protezioni:

- La **riservatezza** dei Refresh Token DEVE essere garantita in transito e archiviazione.
- Per la trasmissione dei token, DEVONO essere utilizzate connessioni **protette da TLS**.
- I Refresh Token DEVONO essere **non indovinabili e sicuri da modifiche**.
- Gli Authorization Server DEVONO implementare il seguente meccanismo per **rilevare gli attacchi di replay**:
  - **Sender-Constrained Tokens**: Vincolare crittograficamente il Refresh Token all'Istanza del Wallet secondo **RFC 9449**. Gli Access Token e i Refresh Token DEVONO essere vincolati alla stessa chiave DPoP. La *DPoP proof* del Refresh Token è richiesta per aggiornare un Access Token. La stessa chiave DPoP DEVE essere utilizzata per generare le *DPoP proof* dell'Access Token in tutte le *Credential Request*.
- **Limitazione dell'uso del Refresh Token**: Come specificato in OPENID4VC-HAIP: "Credential Issuers should be mindful of how long the usage of the refresh token is allowed to refresh a Credential, as opposed to starting the issuance flow from the beginning. For example, if the User is trying to refresh a Credential more than a year after its original issuance, the usage of the refresh tokens is NOT RECOMMENDED." In questa specifica, un nuovo Attestato Elettronico ottenuto eseguendo il Re-issuance Flow DOVREBBE avere la stessa scadenza di quello aggiornato. Pertanto, questa specifica non consente l'aggiornamento infinito dell'Attestato Elettronico con un Refresh Token. Una volta che un Attestato Elettronico scade, l'Utente DEVE completare nuovamente l'intero processo di emissione per ottenere un nuovo Attestato Elettronico. Questa specifica raccomanda di impostare una durata di scadenza del Refresh Token, in base alla sensibilità del grant associato.

Nota

Attestati di Unità di Wallet e DPoP di breve durata: Seguendo la bozza di specifica *OAuth 2.0 Attestation Based Client Authentication* (OAUTH-ATTESTATION-CLIENT-AUTH), l'Authorization Server DEVE associare il Refresh Token alla Client Instance. Per dimostrare questa associazione, la Client Instance DEVE utilizzare il meccanismo di Client Attestation quando aggiorna l'Access Token e la Client Instance DEVE utilizzare la stessa chiave che è stata presentata nel claim `cnf.jwk` della Client Attestation che è stata utilizzata quando il Refresh Token è stato emesso. Tuttavia, ciò richiede che tutti le Client Attestation emesse DEVONO essere associati alla stessa chiave, aprendo così a problemi di non collegabilità. In questa specifica, sia OAUTH-ATTESTATION-CLIENT-AUTH che *OAuth 2.0 Demonstrating Proof of Possession (DPoP)* (**RFC 9449**) DEVONO essere utilizzati. L'uso di DPoP garantisce l'associazione del Refresh Token con la Client Instance come indicato nella sezione 5 di **RFC 9449** *"the Refresh Token MUST be bound to the respective public key [...] a Client MUST present a DPoP proof for the same key that was used to obtain the Refresh Token each time that Refresh Token is used to obtain a new Access Token"*. DPoP garantisce che il Refresh Token sia associato all'Istanza del Wallet (*WP_068b* (pagina 790)). DPoP ensures that the Refresh Token is bound to the Wallet Instance.

Re-issuance Flow

La riemissione comporta la sostituzione degli Attestati Elettronici già memorizzati in un'Istanza del Wallet con nuovi dello stesso tipo di documento. I nuovi Attestati Elettronici DEVONO essere emessi dagli stessi Credential Issuer che hanno originariamente fornito quelli esistenti alla stessa Istanza del Wallet.

Per facilitare questo, in particolare in scenari in cui l'autenticazione dell'Utente non è strettamente richiesta, PUÒ essere utilizzato un Refresh Token Flow (RT) (vedi Sezione *Refresh Token Flow* (pagina 469) per maggiori dettagli). Un Access Token ottenuto come risultato di un Refresh Token Flow NON DEVE essere utilizzato per emettere un Attestato Elettronico che non è presente nell'Istanza del Wallet (prima emissione). Il meccanismo del Refresh Token consente la sostituzione automatica degli Attestati Elettronici, semplificando il processo sia per il Credential Issuer che per l'Utente.

Il Re-issuance Flow delineato in questa sezione è limitato ai seguenti scenari:

- Aggiornamento tecnico del modello/formato dei dati;
- Aggiornamento dell'insieme di attributi dell'Utente.

Questo flusso non si applica alla migrazione di dispositivo o al ripristino da backup. In tali casi l'Istanza del Wallet DEVE eseguire un Wallet-Initiated Authorization Code Issuance Flow

come definito nella Sezione *Issuance Flow* (pagina 447) (vedere Sezione *Flusso di ripristino per Credenziale con associazione hardware* (pagina 324)).

Nel primo caso, l'insieme di attributi dell'Utente del nuovo Attestato Elettronico corrisponderà a quello originale. Ad esempio, un Credential Issuer potrebbe dover aggiornare i Metadata dell'Attestato Elettronico o il formato dei dati senza modificare l'insieme di attributi dell'Utente. In questo caso, il coinvolgimento diretto dell'Utente non è obbligatorio per la sostituzione e l'archiviazione di un Attestato Elettronico.

Nel secondo caso, i Credential Issuer potrebbero anche dover modificare uno o più valori degli attributi dell'Utente durante la riemissione. In questo caso, l'Istanza del Wallet DEVE informare l'Utente che l'insieme di dati degli attributi è stato modificato e DEVE quindi richiedere l'autorizzazione dell'Utente per memorizzare il nuovo Attestato Elettronico.

In entrambi i casi, l'Attestato Elettronico appena emesso DEVE avere la stessa data di scadenza di quello precedente.

La riemissione dopo la scadenza dell'Attestato Elettronico DEVE sempre richiedere l'autenticazione dell'Utente.

Il seguente diagramma descrive il Re-issuance Flow dell'Attestato Elettronico.

Passo 1: Il flusso inizia quando l'Utente apre l'Istanza del Wallet: questo passaggio PUÒ essere attivato da una notifica inviata dal Credential Issuer (utilizzando ad esempio uno dei contatti di comunicazione out-of-band registrati durante il flusso di emissione).

Passo 2: L'Istanza del Wallet DEVE controllare lo stato di ogni Attestato Elettronico memorizzato, recuperando — se non disponibile — un Token di Stato valido (seguendo il flusso descritto nella Sezione *Token di Status List* (pagina 437)) come da (*WP_069* (pagina 790)). L'Istanza del Wallet DEVE poi verificare (*WP_070* (pagina 790)) se un Attestato Elettronico ha lo stato impostato su `0x03 - UPDATE` o `0x0F - ATTRIBUTE_UPDATE`.

Se le condizioni non sono soddisfatte, il flusso DEVE essere interrotto.

Altrimenti, l'Istanza del Wallet DEVE verificare i relativi Access Token. Se sono ancora validi, il Passo 3 PUÒ essere saltato (*WP_071* (pagina 790)).

Passo 3: Se l'Access Token è scaduto e l'Istanza del Wallet ha ancora un Refresh Token valido, l'Istanza del Wallet DEVE ottenere un nuovo Access Token avviando un Refresh Token Flow, secondo la Sezione *Refresh Token Flow* (pagina 469) (*WP_071a* (pagina 790)). Il Refresh Token Flow consente all'Istanza del Wallet di ottenere un nuovo Refresh Token e un nuovo Access Token DPoP per aggiornare l'Attestato Elettronico. Se il Refresh Token è scaduto, è necessario un nuovo flusso di emissione che autentichi l'Utente (*WP_067* (pagina 790) e *WP_071b* (pagina 790)).

Passo 4: L'Istanza del Wallet DEVE utilizzare un Access Token DPoP valido per recuperare il nuovo Attestato Elettronico, richiedendolo al Credential Endpoint seguendo i passi da 12 a 22 della Figura 9 nella Sezione *Issuance Flow* (pagina 447) (*WP_072* (pagina 790)). Quando il

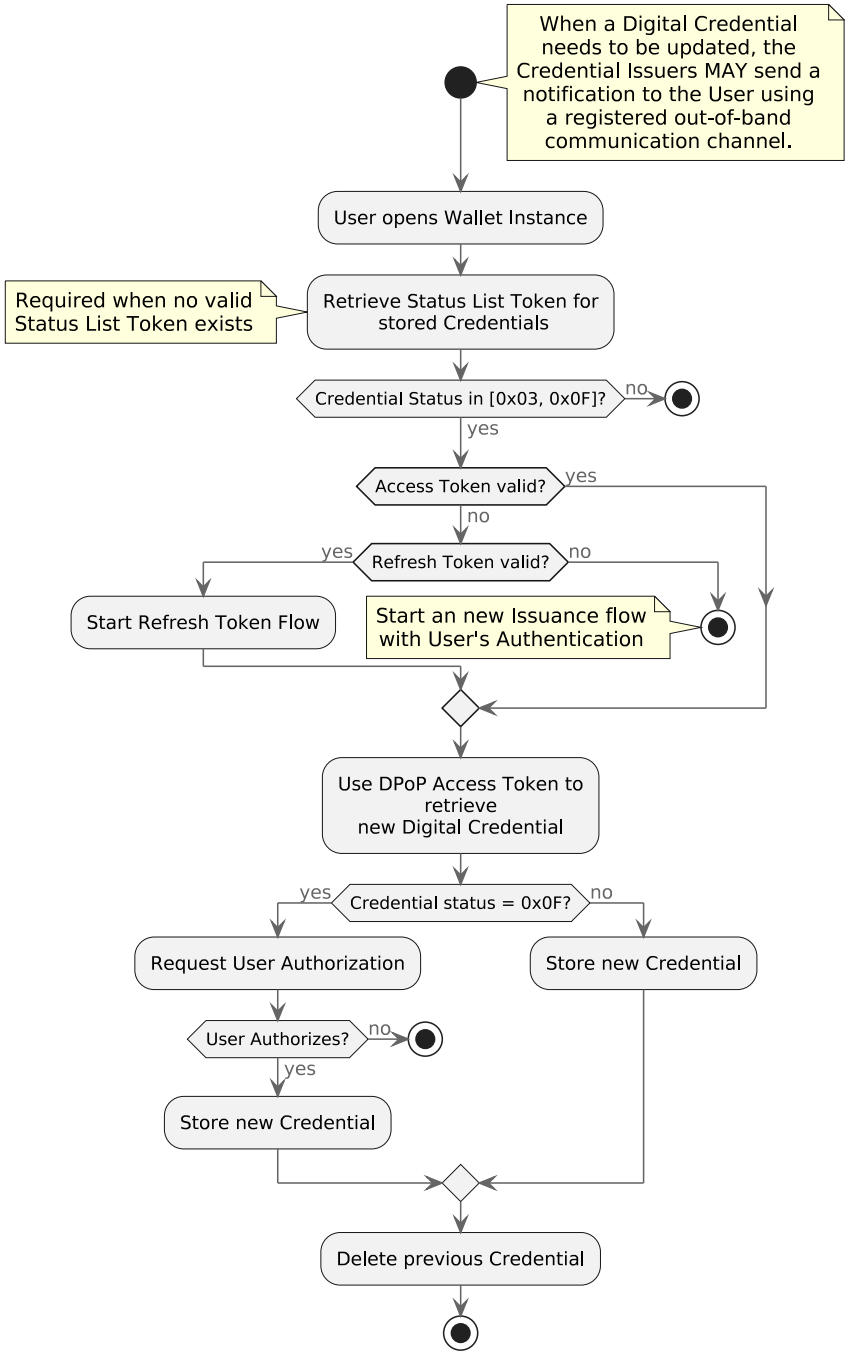


Fig. 12.7: Re-issuance Flow.

nuovo Attestato Elettronico è memorizzato con successo nel secure storage, l'Istanza del Wallet DEVE eliminare quello precedente (*WP_073* (pagina 790)).

Nota

Indipendentemente dal meccanismo di revoca supportato, se lo stato dell'Attestato Elettronico è impostato su `ATTRIBUTE_UPDATE`, l'insieme di attributi dell'Utente nell'Attestato aggiornato non corrisponde a quello memorizzato. In questo caso, l'Istanza del Wallet DEVE richiedere l'autorizzazione dell'Utente per memorizzare il nuovo Attestato aggiornato (*WP_074* (pagina 790)).

Se invece lo stato è impostato su `UPDATE`, solo i parametri dei Metadata sono cambiati. In questo caso, l'Istanza del Wallet DOVREBBE memorizzare il nuovo Attestato senza richiedere autorizzazione o consenso esplicito dell'Utente (*WP_075* (pagina 790)).

Re-issuance Flow: Considerazioni di Sicurezza

Per garantire l'integrità e la sicurezza del Re-issuance Flow, si applicano le seguenti considerazioni di sicurezza.

- Limitazioni dell'Access Token: Un Access Token ottenuto come risultato di un Refresh Token Flow NON DEVE essere utilizzato per l'emissione per la prima volta di un Attestato Elettronico. Ciò garantisce che vengano aggiornati solo gli Attestati Elettronici esistenti nell'Istanza del Wallet.
- Scadenza dell'Attestato Elettronico: Il Credential Issuer DEVE impostare per l'Attestato Elettronico riemesso la stessa data di scadenza del precedente. Ciò impedisce rinnovi indefiniti dell'Attestato Elettronico senza una corretta autenticazione dell'Utente.
- Consenso dell'Utente: Per i Re-issuance Flow attivati da modifiche agli attributi, il consenso dell'Utente DEVE essere ottenuto prima di memorizzare il nuovo Attestato Elettronico.
- Refresh Token vincolato al mittente: I Refresh Token DEVONO essere crittograficamente vincolati all'Istanza del Wallet utilizzando il protocollo DPoP. Ciò mitiga il rischio di uso improprio del token, garantendo che solo l'Istanza del Wallet prevista possa utilizzarlo.

Nota

Durante ogni transazione di emissione di un Attestato Elettronico e, ove supportato, ogni transazione di riemissione/aggiornamento eseguita tramite i flussi descritti in questa sezione, l'Istanza del Wallet DEVE creare e mantenere un corrispondente record di transazione nel

registro delle transazioni dell'Istanza del Wallet (vedi *Dashboard dell'Istanza del Wallet e Registrazione delle Transazioni* (pagina 319)).

L'Istanza del Wallet DEVE creare un record di transazione per ogni processo di emissione. Il record DEVE acquisire le informazioni di contesto rilevanti necessarie a garantire tracciabilità e responsabilizzazione della transazione.

Il record DEVE essere aggiornato man mano che il flusso progredisce per riflettere lo stato della transazione e l'esito (ad es., il/i tipo/i di Attestato emesso/i e la quantità). Nei casi in cui la transazione non si completi con successo, il record DEVE indicare la corrispondente motivazione della mancata conclusione.

Flusso Credential Offer

Un Credential Issuer o una terza parte (ad esempio, Authentic Source, Registro, Catalogo) avvia l'emissione di Attestati Elettronici inviando una Credential Offer all'Istanza del Wallet.

Per richiamare l'Istanza del Wallet corretta, è necessario sapere quale Istanza del Wallet è installata sul dispositivo dell'Utente e quale l'Utente desidera utilizzare. Queste informazioni DOVREBBERO essere ottenute utilizzando la Selection Page descritta in *Design dell'Esperienza Utente* (pagina 39).

- Se la Selection Page è disponibile, l'utente seleziona l'Istanza del Wallet, quindi il Credential Issuer o una terza parte recupera i Wallet metadata come descritto in *Flusso di Recupero dei Wallet Metadata* (pagina 570). Il meccanismo di invocazione dell'Istanza del Wallet dipende dal parametro `credential_offer_endpoint` nei Wallet metadata:
- Se `credential_offer_endpoint` è disponibile e contiene un URL HTTPS (Universal Link), il Credential Issuer o la terza parte DOVREBBE utilizzare quell'endpoint.
- Altrimenti, il Credential Issuer o la terza parte DEVE utilizzare uno degli schemi URL personalizzati: `openid-credential-offer://` (come definito nella Sezione 4 di [Open ID4VCI]) o `haip-vci://` (come definito nella Sezione 4.2 di [OPENID4VC-HAIP]). L'Istanza del Wallet DEVE supportare entrambi gli schemi URL personalizzati.
- Nel caso in cui il Credential Issuer o la terza non supporti la Selection Page o il recupero dei Wallet metadata fallisce per qualche motivo, il Credential Issuer o la terza richiamerà l'Istanza del Wallet utilizzando uno degli schemi URL personalizzati descritti precedentemente.

La Credential Offer può essere trasmessa per valore o per riferimento:

- **Per valore** (parametro `credential_offer`): l'oggetto Credential Offer è incorporato direttamente nell'URI come stringa codificata JSON.

- **Per riferimento** (parametro `credential_offer_uri`): l'URI punta a una risorsa ospitata dal Credential Issuer o dalla terza parte. L'Istanza del Wallet recupera l'oggetto Credential Offer inviando una richiesta HTTP GET a quell'URI.

L'oggetto Credential Offer è un oggetto JSON contenente i parametri definiti nella Sezione 4.1.1 di [OpenID4VCI]. La tabella seguente specifica i parametri con requisiti specifici IT-Wallet:

Claim	Descrizione	Riferimento
credential_issuer	DEVE essere impostato con un URL HTTPS che identifica univocamente il Credential Issuer. Il Wallet utilizza questo valore del parametro per ottenere i metadata del Credential Issuer.	Sezione 4.1.1 di [OpenID4VCI].
credential_configuration_ids	Array di stringhe, ciascuna delle quali specifica un identificatore univoco dell'Attestato Elettronico descritto nella mappa `credential_configurations_supported` nei Metadata del Credential Issuer (*WP_050b* (pagina 790)).	Sezione 4.1.1 di [OpenID4VCI].

continues on next page

Tabella 12.1 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
grants	OBBLIGATORIO. DEVE contenere l'oggetto `authorization_code` con i seguenti parametri: • **issuer_state**: OPZIONALE. Stringa opaca utilizzata per associare la successiva Authorization Request con il Credential Isser. PUO' essere associata a un determinato Credential Dataset fornito da una specifica Fonte Autentica. Il Wallet DEVE includerlo nella successiva Authorization Request quando presente. Deve essere un'URN e contenere le seguenti informazioni: – *authenticSourceId*: OBBLIGATORIO. DEVE corrispondere al valore `entity_id` della Fonte Autentica che fornisce i Credential Dataset, come indicato nel *Registro delle Fonti Autentiche* (pagina 173). – *datasetId*: OBBLIGATORIO. Identificativo univoco del dataset fornito dalla Fonte Autentica, come indicato nel *Registro delle Fonti Autentiche* (pagina 173). – *objectId*: OPZIONALE. Identificativo univoco del Credential Dataset disponibile presso la Fonte Autentica. Il parametro `issuer_state` DEVE seguire la seguente struttura: `urn:it-wallet:credential-offer:{authenticSourceId}:{datasetId}` se `objectId` è assente oppure `urn:it-wallet:credential-offer:{authenticSourceId}:{datasetId}:{objectId}` nel caso in cui `objectId` è presente. Il segmento opzionale `objectId` DEVE essere omesso quando non disponibile; NON DEVE essere utilizzato un segmento finale vuoto. Il valore di questo URN DEVE essere cifrato utilizzando la chiave pubblica PDND relativa al Consumer dell'e-service `GetAttributeClaims`. • **authorization_server**: OBBLIGATORIO quando il Credential Issuer utilizza più di un authorization server nella sua soluzione. Stringa che identifica l'Authorization Server da utilizzare. Il valore DEVE corrispondere a uno dei valori mappati nell'array `authorization_servers` dei metadata del Credential Issuer. NON DEVE essere utilizzato se `authorization_servers` è assente o non ha voci multiple.	Sezione 4.1.1 di [OpenID4VCI] e Sezione 4.1 di [OPENID4VC-HAIP].

Tabella 12.1 – continua dalla pagina precedente

Claim	Descrizione	Riferimento

Nota

Quando si utilizza `credential_offer_uri` (per riferimento), il Credential Issuer o la terza parte DOVREBBE utilizzare un URI univoco per ogni Credential Offer o altrimenti prevenire il caching dell'URI, come raccomandato nella Sezione 4.1.3 di [OpenID4VCI].

Esempi non normativi

Esempio 1: Credential Offer per valore

La Credential Offer può essere trasmessa per valore utilizzando qualsiasi metodo di invocazione supportato (schema URL personalizzato o Universal Link):

```
openid-credential-offer://?credential_offer=%7B%22credential_issuer%22%3A%22https%3A//credenti
al-issuer.example.org%22%2C%22credential_configuration_ids%22%3A%5B%22dc_sd_jwt_Education_degr
ee%22%5D%2C%22grants%22%3A%7B%22authorization_code%22%3A%7B%22issuer_state%22%3A%22eyJhbGciOiJ
SU0Et...F77QK8%22%7D%7D%7D
```

L'oggetto Credential Offer decodificato:

```
{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["dc_sd_jwt_Education_degree"],
  "grants": {
    "authorization_code": {
      "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

Esempio 2: Credential Offer per riferimento da Credential Issuer

Il codice QR o il pulsante href contiene:

```
openid-credential-offer://?credential_offer_uri=https%3A%2F%2Fcredential-issuer.example.org%2F
offers%2F8f3a2b1c
```

L'Istanza del Wallet invia una richiesta HTTP GET:

```
GET /offers/8f3a2b1c HTTP/1.1
Host: credential-issuer.example.org
Accept: application/json
```

Il Credential Issuer risponde:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["dc_sd_jwt_EuropeanDisabilityCard"],
  "grants": {
    "authorization_code": {
      "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

Esempio 3: Credential Offer per riferimento da terza parte

Una terza parte, come un'Authentic Source, genera una Credential Offer per un Credential Issuer.

Il codice QR contiene:

```
openid-credential-offer://?credential_offer_uri=https%3A%2F%2Fauthentic-source.gov.example%2Fc
redential-offers%2Fabc123
```

L'Istanza del Wallet invia:

```
GET /credential-offers/abc123 HTTP/1.1
Host: authentic-source.gov.example
Accept: application/json
```

L'Authentic Source risponde:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "credential_issuer": "https://credential-issuer.example.org",
  "credential_configuration_ids": ["mso_mdoc_mDL"],
  "grants": {
    "authorization_code": {
     "issuer_state": "eyJhbGciOiJSU0Et...F77QK8",
    }
  }
}
```

12.1.3 Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID

Questa Sezione definisce un protocollo di Autenticazione eID Substantial con Verifica MRTD che si integra nel flusso di emissione dell'IT-Wallet ID come definito nella Sezione *Flussi Dettagliati per l'Emissione di Attestati Elettronici* (pagina 447) estendendo i flussi OAuth 2.0 per includere:

- Identificazione elettronica con Livello di Garanzia "Substantial" (LoA3) come step di autenticazione primaria.
- Verifica di documento elettronico come livello aggiuntivo di verifica dell'identità.
- Correlazione di sessione e binding di sicurezza tra gli step di autenticazione.
- Integrazione con i flussi di emissione Attestati Elettronici.

Mentre l'autenticazione CIEid con LoA High rimane il metodo primario per l'attivazione Wallet e l'emissione dell'IT-Wallet ID, il meccanismo di Autenticazione eID Substantial con Verifica MRTD definito in questa Sezione fornisce un approccio alternativo per migliorare l'accessibilità e l'usabilità del servizio, senza compromettere la sicurezza complessiva dell'ecosistema IT-Wallet.

Nota

Questa Sezione attualmente supporta solo la carta d'identità CIE per il protocollo di verifica MRTD, ma il protocollo descritto in questa Sezione PUÒ essere esteso per supportare altri Documenti MRTD come i Passaporti Elettronici.

Principi di Progettazione

Il protocollo aderisce ai seguenti principi di progettazione:

- **Compatibilità con Standard**: Estende i framework di autorizzazione esistenti senza modifiche che rompano la compatibilità con il framework OAuth Authorization standard.
- **Autenticazione Multi-Step**: Implementa l'applicazione progressiva dell'autenticazione con verifica MRTD.
- **Integrità di Sessione**: Mantiene correlazione sicura di sessione attraverso gli step di autenticazione.
- **Flusso Unificato**: Integra fattori di autenticazione multipli in una singola sessione di autorizzazione.

Architettura di Sistema

L'architettura di sistema comprende i seguenti componenti principali:

- **Istanza del Wallet:** Gestisce la richiesta IT-Wallet ID secondo le Specifiche IT-Wallet, supportando capacità crittografiche aggiuntive per la lettura di Documenti Elettronici MRTD/IAS secondo le specifiche ICAO 9303 e BSI-03110.
- **Sistema di Autenticazione eID Substantial con Verifica MRTD:** Orchestra il flusso di autenticazione, integrando Provider di Identità LoA3, Servizio di Verifica di Documento Elettronico, ed eseguendo tutti i controlli di correlazione di identità per garantire che l'Utente che richiede un IT-Wallet ID sia lo stesso Utente autenticato.
  - **Server di Autorizzazione EAA:** Gestisce il flusso di autorizzazione per l'Emissione IT-Wallet ID, coordinando l'autenticazione Utente e l'identity proofing remoto.
  - **Servizio MRTD PoP:** Gestisce il proof of possession di documento elettronico con validazione crittografica del documento.
- **Provider di Identità LoA3:** Fornisce Schemi di Identificazione Elettronica con conformità eIDAS LoA3 (CIEid e SPID).
- **Registro Nazionale CIE:** Agisce come fonte autentica per la CIE. Fornisce informazioni relative allo stato di validità del documento. Fornisce anche opzionalmente i dati MRZ (numero documento, data di nascita, data di scadenza, cittadinanza e genere) per migliorare l'esperienza utente.

Flusso High-Level

Il protocollo consiste nelle seguenti fasi sequenziali:

1. **Richiesta di Autorizzazione OAuth**: PAR e richiesta di autorizzazione con requisiti di Autenticazione eID Substantial con Verifica MRTD.
2. **Autenticazione Primaria**: Identificazione elettronica LoA3 (SPID/CIEid L2).
3. **Validazione MRTD Proof of Possession (PoP)**: Lettura di documento elettronico e verifica crittografica (controlli di Proof of Possession, integrità e autenticità).
4. **Risposta di Autorizzazione OAuth**: Emissione finale del codice di autorizzazione.

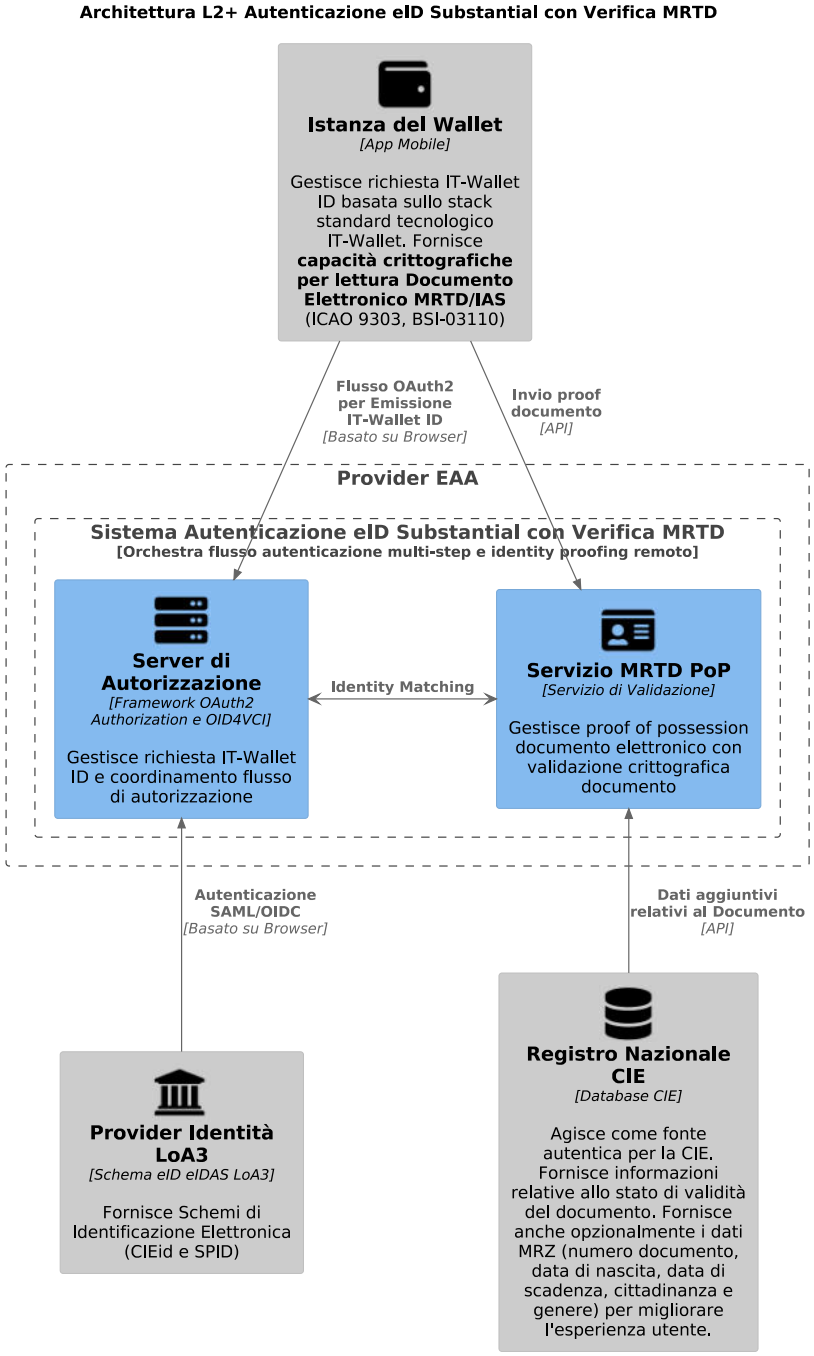


Fig. 12.8: Architettura del Sistema di Autenticazione eID Substantial con Verifica MRTD.

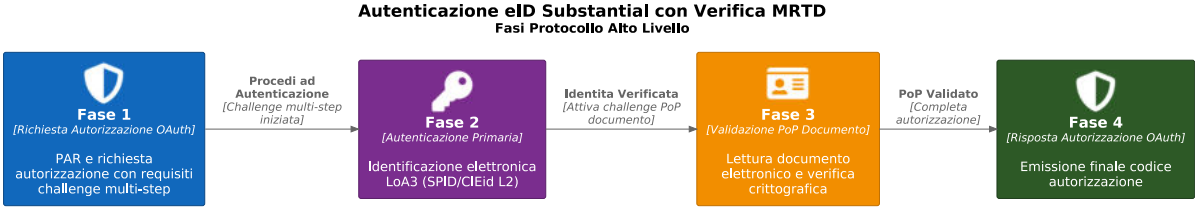


Fig. 12.9: Flusso High-Level di Autenticazione eID Substantial con Verifica MRTD.

Gestione Sessione

Il Server di Autorizzazione DEVE mantenere una sessione unificata attraverso tutti gli step di autenticazione, assicurando:

- **Continuità di Sessione**: Identificatore di sessione singolo durante tutto il flusso di autenticazione.
- **Correlazione di Stato**: I risultati di autenticazione da ogni step sono correlati.
- **Binding di Sicurezza**: Binding tra gli step di autenticazione.

Poiché il Server di Autorizzazione EAA e il Servizio MRTD PoP operano all'interno dell'implementazione dei confini del Provider EAA, è RACCOMANDATO che la sessione OAuth 2.0 e i nonce utilizzati nel flusso del protocollo siano gestiti correttamente da entrambi i componenti.

Quando entrambi i servizi operano all'interno dello stesso confine di fiducia, i seguenti meccanismi sono disponibili per la correlazione di sessione:

- Accesso diretto alla memoria agli store di sessione condivisi.
- Autenticazione interna service-to-service utilizzando chiavi pre-condivise.
- Validazione nonce sincronizzata senza overhead di comunicazione esterna.
- Logging di audit unificato e correlazione di eventi di sicurezza.

Quando il Server di Autorizzazione EAA o il Servizio MRTD PoP sono erogati al di fuori dell'implementazione dei confini del Provider EAA, le *Considerazioni di Sicurezza* (pagina 501) DEVONO essere prese in considerazione per rafforzare la gestione delle sessioni di autenticazione Utente. Queste misure includono ma non sono limitate alla gestione sicura dei token di sessione, validazione di sessione distribuita, e meccanismi di audit trail migliorati.

Flusso Low-Level

Questa sezione fornisce dettagli tecnici sull'Emissione dell'IT-Wallet ID utilizzando Livello di Garanzia Substantial e identity proofing remoto attraverso verifica del documento elettronico utilizzando un servizio di Verifica MRTD.

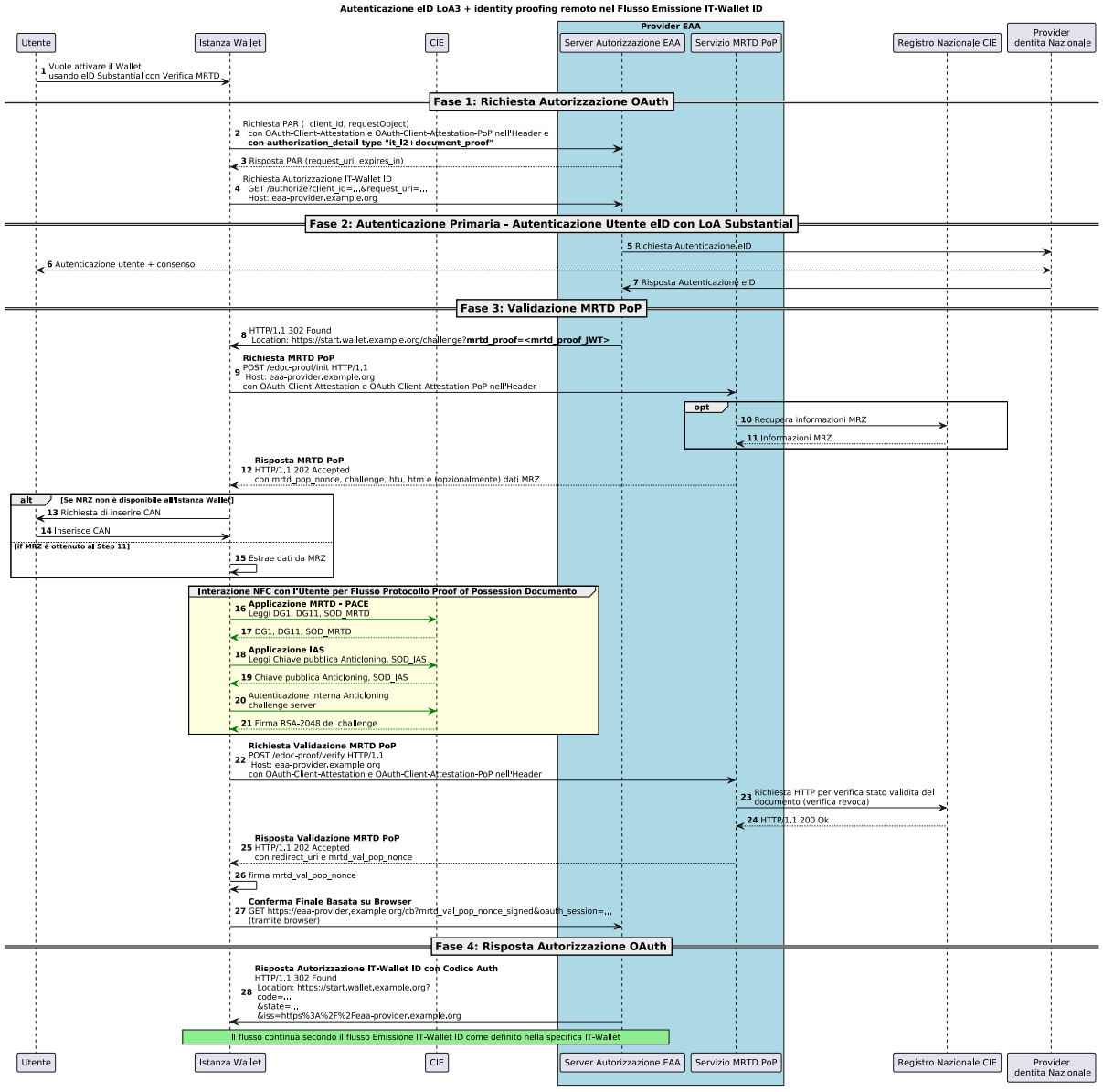


Fig. 12.10: Flusso Dettagliato di Autenticazione eID Substantial con Verifica MRTD.

Fase 1: Richiesta di Autorizzazione OAuth

L'Istanza del Wallet inizia il flusso di Autenticazione eID Substantial con Verifica MRTD inviando una Pushed Authorization Request (PAR) contenente una JWT-secured Authorization Request (JAR) con requisiti di autenticazione specificati tramite Rich Authorization Requests (RAR).

Authorization Details

Il JWT Request Object DEVE contenere gli stessi parametri come definiti in credential-issuance-endpoint:Pushed Authorization Request Endpoint. L'inclusione dell'oggetto `it_l2+document_proof` in `authorization_details` e il metodo di autenticazione applicato dall'*Authorization Server* sono definiti in credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente (*WP_158* (pagina 781), *CI_199* (pagina 825)).

Di seguito un esempio non normativo di PAR:

```
POST /as/par HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

client_id=47b982369791d08003a7283f059cb0d1&
request=eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWF1dGh6LXJlcStqd3QiLCJraWQiOiJiMDFiODIwOGQ5ZTZjYzg
zNGQ4N2RjMzU2YWI1MDE3MCJ9.eyJqdGkiOiJmODU1NWNlYi1jNjVjLTQwMjUtOTM3OC1iNjY3MmI2MTQ5YWYiLCJhdWQiO
iJodHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZyIsImlhdCI6MTcxNTg0MjU2MCwiZXhwIjoxNzE1ODQyODYwLCJy
ZXNwb25zZV90eXBlIjoiY29kZSIsInJlc3BvbnNlX21vZGUiOiJmb3JtX3Bvc3Quand0IiwiY2xpZW50X2lkIjoiNDdiOTg
yMzY5NzkxZDA4MDAzYTcyODNmMDU5Y2IwZDEiLCJpc3MiOiI0N2I5ODIzNjk3OTFkMDgwMDNhNzI4M2YwNTljYjBkMSIsIn
N0YXRlIjoiZnlaaU9MOUxmMkNlS3VOVDJKenhpTFJEaW5rMHVQY2QiLCJjb2RlX2NoYWxsZW5nZSI6IkU5TWVsaG9hMk93d
kZyRU1USmd1Q0hhb2VLMXQ4VVJXYnVHSlNzdFctY00iLCJjb2RlX2NoYWxsZW5nZV9tZXRob2QiOiJTMjU2Iiwic2NvcGUi
OiJlaWQiLCJhdXRob3JpemF0aW9uX2RldGFpbHMiOlt7InR5cGUiOiJvcGVuaWRfY3JlZGVudGlhbCIsImNyZWRlbnRpYWx
fY29uZmlndXJhdGlvbl9pZCI6ImRjX3NkX2p3dF9laWQifSx7InR5cGUiOiJpdF9sMitkb2N1bWVudF9wcm9vZiIsImNoYW
xsZW5nZV9tZXRob2QiOiJtcnRkK2lhcyIsImlkcGhpbnRpbmciOiJodHRwczovL2lkcC5leGFtcGxlLm9yZyIsImNoYWxsZ
W5nZV9yZWRpcmVjdF91cmkiOiJodHRwczovL3N0YXJ0LndhbGxldC5leGFtcGxlLm9yZy9jaGFsbGVuZ2UifV0sInJlZGly
ZWN0X3VyaSI6Imh0dHBzOi8vc3RhcnQud2FsbGV0LmV4YW1wbGUub3JnIn0.AuthRequestSign456_NoKidJWTSignatur
e-abc123def456ghi789jkl012mno345pqr678stu901vwx234yz567
```

La Risposta PAR e la Richiesta di Autorizzazione sono le stesse delle Specifiche IT-Wallet.

Fase 2: Autenticazione Primaria

Dopo l'elaborazione con successo del PAR, il Server di Autorizzazione reindirizza l'User Agent al Provider di Identità per l'autenticazione primaria, secondo credential-issuance-endpoint: Selezione del Metodo di Autenticazione dell'Utente. L'Utente completa il flusso di autenticazione e il Server di Autorizzazione correla l'identità autenticata con la sessione OAuth attiva.

In questo flusso **LoA3** (ISO/IEC 29115) corrisponde a eIDAS **Substantial** e ai mezzi nazionali L2 (SPID L2 / CieID Substantial). **LoA High** corrisponde a eIDAS High e a CieID LoA High (CIE L3). Un'autenticazione Substantial NON DEVE essere registrata o trattata come High (*CI_205* (pagina 825)).

Il Server di Autorizzazione EAA DEVE assicurare che il parametro `mrtd_auth_session` sia mantenuto durante questa fase per la correlazione appropriata di sessione con gli step di autenticazione successivi.

Quando gli attributi MRTD diventano disponibili (Fase 3), il Provider EAA DEVE vincolare l'identità verificata dall'IdP con l'identità nell'MRTD confrontando almeno il codice fiscale o un identificativo nazionale univoco equivalente presente in entrambe le fonti. Una mancata corrispondenza DEVE interrompere l'emissione (*WP_159* (pagina 781), *CI_200* (pagina 825)).

Nota

Nel caso in cui l'Utente dovesse eseguire un'autenticazione LoA High, la successiva fase 3 DEVE essere saltata.

Fase 3: Flusso di Validazione MRTD PoP

Dopo l'autenticazione primaria con successo, il Server di Autorizzazione inizia il flusso di proof of possession del documento elettronico fornendo all'Istanza del Wallet i parametri necessari per la validazione MRTD.

JWT di Prova MRTD

Il Server di Autorizzazione DEVE fornire un JWT firmato contenente i requisiti di challenge per la validazione del documento. La struttura JWT è definita come segue:

Tabella 12.2: Header JWT di Prova MRTD

Parametro	Tipo	Descrizione
alg	string	OBBLIGATORIO. Algoritmo di firma.
typ	string	OBBLIGATORIO. DEVE essere `mrtd-ias+jwt`.
kid	string	OBBLIGATORIO. Identificatore della chiave del Provider EAA che DEVE essere utilizzata per verificare la firma di questo JWT.

Tabella 12.3: Payload JWT di Prova MRTD

Parametro	Tipo	Descrizione
iss	string	OBBLIGATORIO. Identificatore Provider EAA.
aud	string	OBBLIGATORIO. Identificatore Istanza del Wallet.
iat	integer	OBBLIGATORIO. Tempo di emissione (Unix timestamp).
exp	integer	OBBLIGATORIO. Tempo di scadenza (Unix timestamp).
status	string	OBBLIGATORIO. Stato del challenge. DEVE essere `require_interaction`.
type	string	OBBLIGATORIO. Tipo di interazione richiesta. DEVE essere impostato a `mrtd+ias`.
mrtd_auth_session	string	OBBLIGATORIO. Identificatore di sessione.
state	string	OBBLIGATORIO. DEVE essere lo stesso valore del Request Object iniziale.
mrtd_pop_jwt_nonce	string	OBBLIGATORIO. Valore nonce per protezione replay. DEVE essere ottenuto dal Servizio MRTD PoP che DEVE avere controllo diretto su questo valore.
htu	string	OBBLIGATORIO. URI HTTP dell'endpoint di inizializzazione MRTD PoP.
htm	string	OBBLIGATORIO. Metodo HTTP per la richiesta di inizializzazione MRTD PoP. DEVE essere `POST`.

Un esempio non normativo è riportato di seguito:

```
HTTP/1.1 302 Found
Location: https://start.wallet.example.org/challenge?challenge_info=eyJhbGciOiJFUzI1NiIsInR5cCI
6Im1ydGQtaWFzK2p3dCIsImtpZCI6ImQ0ZjhlMDJlZWJhMDdhNTM3MmUxOGVjYzU4NzA5ZDA2In0.eyJpc3MiOiJodHRwcz
ovL3BpZC1wcm92aWRlci5leGFtcGxlLm9yZyIsImF1ZCI6IjQ3Yjk4MjM2OTc5MWQwODAwM2E3MjgzZjA1OWNiMGQxIiwia
WF0IjoxNzUzNTU1MzU4LCJleHAiOjE3NTM1NTU2NTgsInN0YXR1cyI6InJlcXVpcmVfaW50ZXJhY3Rpb24iLCJ0eXBlIjoi
bXJ0ZCtpYXMiLCJtcnRkX2F1dGhfc2Vzc2lvbiI6Ind4cm9WckJZMk1DcTRkRE5HWEFDUyIsInN0YXRlIjoiZnlaaU9MOUx
mMkNlS3VOVDJKenhpTFJEaW5rMHVQY2QiLCJtcnRkX3BvcF9qd3Rfbm9uY2UiOiJub25jZTEiLCJodHUiOiJodHRwczovL2
Vkb2MtcHJvb2YvaW5pdCIsImh0bSI6IlBPU1QifQ.i6p_FN7qNNawyL4KnOV1r8FrNVjzd-7Ve1wEGASHNnlXwuJ1f216v0
Ml_KpVrq9yXkmOo_M2xZwih2SlHVfrzkuG3Pn7LWRL7dsyCtqEY2e58rFHjCa2miBnnKr0NU4wcBMMYe2_qKCOkA7SOa7us
NTBluBLMQ28GfiMbr3tcpfpM4rD0POKQcfijvNkNbh-VdOxM8GdHb6IQO_xfpsaSzd8cc0k5yIYCWjDTeINVKebIz4m9Rm2
JStvRrWUq8OCqkv-8dTJH9q-JXx0PzJC998RMwe6tqSL-kkE3dZLWwCJdP8Z7bITtowU49rEe-AkrGxVma4ANPq317umEfU
wmw
```

Il server di autorizzazione DEVE:

- Generare un identificativo univoco della challenge con entropia sufficiente (minimo 128 bit) per la sicurezza crittografica.
- Creare un MRTD Proof JWT con header (`alg`, `typ`, `kid`) e parametri di payload appropriati (come definiti nella tabella sopra).
- Firmare MRTD proof JWT utilizzando la sua chiave privata con l'algoritmo crittografico scelto. Vedere la sezione *Algoritmi Crittografici* (pagina 647).
- Generare una URL di reindirizzamento, che punta ad una universal link dell'istanza del wallet.
- Impostare il timeout di reindirizzamento per evitare attese indefinite e gestire gli scenari di timeout.
- Richiedere opzionalmente le informazioni MRZ direttamente al Registro Nazionale CIE utilizzando il codice identificativo fiscale dell'utente autenticato.
- Mantenere la correlazione di sessione tra il risultato LoA3 e la verifica di verifica del documento.

L'Istanza del Wallet DEVE:

- Validare la firma JWT utilizzando la chiave pubblica del Provider EAA ottenuta tramite valutazione di fiducia.
- Verificare che il claim `aud` corrisponda al suo `client_id`.
- Verificare che i claim `iat` ed `exp` indichino che il token abbia una data di emissione corretta e non sia scaduto.
- Verificare che il campo `status` sia impostato a "require_interaction".
- Verificare che il tipo di autenticazione corrisponda al valore atteso `mrtd+ias`.
- Estrarre HTTP target URI (`htu`) e metodo (`htm`) per lo step successivo.
- Gestire errori di validazione JWT e di rete, e fornire feedback utente con meccanismi di retry appropriati.
- Estrarre i parametri di correlazione (`mrtd_auth_session`, `state`, `mrtd_pop_jwt_nonce`) per le richieste successive.

Richiesta MRTD PoP

L'Istanza del Wallet DEVE inviare una Richiesta HTTP POST all'endpoint di inizializzazione del Servizio MRTD PoP con `application/json` come content type, includendo Wallet Attestation e Wallet Attestation JWT PoP nell'header secondo OAUTH-ATTESTATION-CLIENT-AUTH. Il payload della Richiesta contiene i seguenti parametri:

Tabella 12.4: Parametri Richiesta MRTD PoP

Parametro	Tipo	Descrizione
mrtd_auth_session	string	OBBLIGATORIO. Identificatore di sessione per binding di sessione.
mrtd_pop_jwt_nonce	string	OBBLIGATORIO. Valore nonce ottenuto dal JWT di Prova MRTD.

Di seguito un esempio non normativo di una Richiesta MRTD PoP:

```
POST /edoc-proof/init HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

{
  "mrtd_auth_session": "wxroVrBY2MCq4dDNGXACS",
  "mrtd_pop_jwt_nonce": "f42cccb7f1c8269f9d4aeefe339c6b13"
}
```

L'Istanza del Wallet DEVE:

- Validare la firma JWT di Prova MRTD utilizzando la chiave pubblica del Provider EAA.
- Verificare che il claim JWT `aud` corrisponda al suo `client_id`.
- Assicurare che il claim JWT `exp` indichi che il token non è scaduto.
- Estrarre i valori `mrtd_auth_session` e `mrtd_pop_jwt_nonce` per correlazione.
- Includere Wallet Attestation valida secondo OAUTH-ATTESTATION-CLIENT-AUTH.
- Gestire errori di rete e implementare meccanismi di retry appropriati.

Il Servizio MRTD PoP DEVE:

- Validare la Wallet Attestation secondo OAUTH-ATTESTATION-CLIENT-AUTH.
- Verificare che il parametro `mrtd_auth_session` corrisponda a una sessione attiva.
- Validare che il parametro `mrtd_pop_jwt_nonce` corrisponda a quello emesso nello step precedente.

Il Servizio MRTD PoP PUÒ richiedere le informazioni MRZ (numero documento, data di nascita, data di scadenza, cittadinanza e genere) direttamente al Registro Nazionale CIE utilizzando il codice fiscale dell'Utente autenticato. In questo caso, il Servizio MRTD PoP è in grado di verificare se l'Utente autenticato possiede una CIE valida e se non è il caso, DEVE inviare una Risposta di Errore HTTP con codice di errore HTTP `access_denied`.

Risposta MRTD PoP

Se la Richiesta HTTP è elaborata con successo, il Servizio MRTD PoP DEVE inviare una Risposta HTTP con codice *202 Accepted* e `application/jwt` come content type. La struttura JWT è definita come segue:

Tabella 12.5: MRTD PoP Response Parameters

Parameter	Type	Description
alg	string	OBBLIGATORIO. Algoritmo di firma.
typ	string	OBBLIGATORIO. DEVE essere `mrtd-ias-pop+jwt`.
kid	string	OBBLIGATORIO. Identificatore della chiave del Provider EAA che DEVE essere utilizzata per verificare la firma di questo JWT.

Tabella 12.6: Parametri Risposta MRTD PoP

Parametro	Tipo	Descrizione
iss	string	OBBLIGATORIO. Identificatore Provider EAA.
aud	string	OBBLIGATORIO. Identificatore Istanza del Wallet.
iat	integer	OBBLIGATORIO. Tempo di emissione (Unix timestamp).
exp	integer	OBBLIGATORIO. Tempo di scadenza (Unix timestamp).
challenge	string	OBBLIGATORIO. Dati di challenge per validazione crittografica del documento. DOVREBBE essere il valore hash SHA-256 di un valore casuale con `mrtd_auth_session` e timestamp per binding crittografico.
mrtd_pop_ nonce	string	OBBLIGATORIO. Valore nonce unico per lo step successivo, prevenendo attacchi replay.
mrz	string	OPZIONALE. Informazioni Machine Readable Zone inclusi numero documento, data di nascita, data di scadenza, cittadinanza e genere.
htu	string	OBBLIGATORIO. URI HTTP dell'endpoint di validazione MRTD PoP.
htm	string	OBBLIGATORIO. Metodo HTTP per la richiesta di validazione MRTD PoP. DEVE essere `POST`.

Di seguito un esempio non normativo di una Risposta MRTD PoP:

```
HTTP/1.1 202 Accepted
Content-Type: application/jwt; charset=utf-8

eyJhbGciOiJFUzI1NiIsInR5cCI6Im1ydGQtaWFzLXBvcCtqd3QiLCJraWQiOiJiM2YxYTZjMmU5ZDU0YThmOWMzZTdkMW
EyZjRiNmM3OCJ9.eyJpc3MiOiJodHRwczovL3BpZC1wcm92aWRlci5leGFtcGxlLm9yZyIsImF1ZCI6Imh0dHBzOi8vd2F
sbGV0LmV4YW1wbGUub3JnL2luc3RhbmNlLzEyMzQ1IiwiaWF0IjoxNzUzNTU1ODAwLCJleHAiOjE3NTM1NTYwMDAsIm1yd
GRfcG9wX25vbmNlIjoiOWYyYzRhN2UzYjFkOGM5YTZlNWY0YjJhMWMzZDdlOGYiLCJtcnoiOiIuLi4iLCJjaGFsbGVuZ2U
iOiIuLi4iLCJodHUiOiIuLi4iLCJodG0iOiIuLi4ifQ.signature
```

Header JWT decodificato:

```
{
  "alg":"ES256",
  "typ":"mrtd-ias-pop+jwt",
  "kid":"b3f1a6c2e9d54a8f9c3e7d1a2f4b6c78"
}
```

Body JWT decodificato:

```
{
  "iss":"https://eaa-provider.example.org",
  "aud":"https://wallet.example.org/instance/12345",
  "iat": 1753555800,
  "exp": 1753556000,
  "mrtd_pop_nonce":"9f2c4a7e3b1d8c9a6e5f4b2a1c3d7e8f",
  "mrz":"...",
  "challenge":"...",
  "htu":"...",
  "htm":"..."
}
```

Il Servizio MRTD PoP DEVE:

- Generare dati di challenge crittograficamente sicuri per validazione `MRTD+IAS` con entropia sufficiente (da utilizzare nel protocollo Anti-Cloning Internal Authentication dall'Istanza del Wallet), memorizzandoli con tempo di scadenza appropriato. Inoltre, il Servizio MRTD PoP DEVE assicurare unicità del challenge per prevenire attacchi di riutilizzo.
- Creare un nuovo `mrtd_pop_nonce` unico per lo step successivo per prevenire attacchi replay.
- Validare continuità di sessione assicurando che il parametro `mrtd_auth_session` corrisponda a una sessione attiva.
- Restituire stato HTTP *202 Accepted* per indicare iniziazione di elaborazione asincrona.
- Includere header Content-Type appropriato (`application/jwt; charset=utf-8`).
- Gestire errori di servizio e restituire risposte di errore appropriate.
- Estrarre e validare informazioni MRZ se fornite da servizi di registro esterni.

L'Istanza del Wallet DEVE:

- Validare stato risposta HTTP (*202 Accepted*) e content type.
- Analizzare risposta JSON e validare parametri richiesti (`challenge`, `mrtd_pop_nonce`).
- Estrarre dati di challenge per validazione crittografica del documento.
- Memorizzare nuovo valore `mrtd_pop_nonce` in modo sicuro per richieste di validazione successive.
- Validare informazioni MRZ opzionali se presenti nella risposta.
- Estrarre HTTP target URI (`htu`) e metodo (`htm`) per lo step successivo.
- Gestire errori, fornendo feedback utente relativo.
- Memorizzare dati di challenge temporaneamente in memoria sicura (non storage persistente).
- Preparare sessione di lettura NFC.

L'Istanza del Wallet esegue lettura e validazione di documento elettronico basata su NFC, poi invia l'evidenza al Provider EAA per verifica finale e correlazione di identità con il risultato di autenticazione LoA3.

Richiesta di Validazione MRTD PoP

Dopo che tutte le evidenze sono state raccolte tramite interazione NFC con il Documento Elettronico, l'Istanza del Wallet DEVE inviare tutti i dati al Server di Autorizzazione del Provider EAA per la validazione finale e controlli incrociati di identità. L'Istanza del Wallet DEVE inviare una Richiesta HTTP POST con `application/json` come content type, includendo Wallet Attestation e Wallet Attestation JWT PoP nell'header secondo OAUTH-ATTESTATION-CLIENT-AUTH. Il payload della Richiesta contiene i seguenti parametri.

Tabella 12.7: Parametri Richiesta di Validazione MRTD PoP

Parametro	Tipo	Descrizione
mrtd_validation_jwt	string	OBBLIGATORIO. JWT contenente evidenza del documento, proof crittografici, Data Groups (DGs), e risultati di validazione.
mrtd_auth_session	string	OBBLIGATORIO. Identificatore di sessione per binding di sessione.
mrtd_pop_nonce	string	OBBLIGATORIO. DEVE corrispondere al valore ottenuto dalla Risposta MRTD PoP.

Struttura JWT di Validazione

La struttura del JWT di Validazione (`mrtd_validation_jwt`) è data nella seguente tabella.

Tabella 12.8: Parametri Header JWT di Validazione

Parametro Header	Tipo	Descrizione
alg	string	OBBLIGATORIO. Algoritmo di firma.
typ	string	OBBLIGATORIO. DEVE essere `mrtd-ias+jwt`.
kid	string	OBBLIGATORIO. Identificatore della chiave dell'Istanza del Wallet che DEVE essere utilizzata per verificare la firma di questo JWT.

Tabella 12.9: Parametri Payload JWT di Validazione

Parametro Payload	Tipo	Descrizione
iss	string	OBBLIGATORIO. Identificatore Istanza del Wallet.
aud	string	OBBLIGATORIO. Identificatore Provider EAA.
iat	integer	OBBLIGATORIO. Tempo di emissione (Unix timestamp).
exp	integer	OBBLIGATORIO. Tempo di scadenza (Unix timestamp).
document_type	string	OBBLIGATORIO. Tipo di documento validato. DEVE essere impostato a `cie`.
mrtd	JSON Object	OBBLIGATORIO. Dati di validazione MRTD contenenti Data Groups e SOD.
ias	JSON Object	OBBLIGATORIO. Dati di validazione IAS contenenti Anti-Cloning Public Key, e SOD.

Struttura Oggetto MRTD

L'oggetto `mrtd` contiene i seguenti campi:

Tabella 12.10: Struttura Oggetto MRTD

Campo	Tipo	Descrizione
dg1	string	OBBLIGATORIO. Data Group 1 codificato Base64 (informazioni MRZ: numero documento, data di nascita, data di scadenza, cittadinanza e genere).
dg11	string	OBBLIGATORIO. Data Group 11 codificato Base64 (dati personali aggiuntivi).
sod_mrtd	string	OBBLIGATORIO. Security Object of Document per MRTD codificato Base64.

Struttura Oggetto IAS

L'oggetto `ias` contiene i seguenti campi:

Tabella 12.11: Struttura Oggetto IAS

Campo	Tipo	Descrizione
ias_pk	string	OBBLIGATORIO. Chiave pubblica IAS codificata Base64 in formato DER.
sod_ias	string	OBBLIGATORIO. Security Object of Document per IAS codificato Base64.
challenge_signed	string	OBBLIGATORIO. Risposta di challenge firmata codificata Base64.

Di seguito un esempio non normativo di una Richiesta di Validazione MRTD PoP:

```
POST /edoc-proof/verify HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
OAuth-Client-Attestation: eyJhbGciOiJFUzI1NiIsImtp…
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUz…

{
  "mrtd_validation_jwt":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3dhbGxldC5le
GFtcGxlLm9yZy9pbnN0YW5jZS8xMjM0NSIsImF1ZCI6Imh0dHBzOi8vcGlkLXByb3ZpZGVyLmV4YW1wbGUub3JnIiwiaWF0
IjoxNzUzNTU1NDAwLCJleHAiOjE3NTM1NTU3MDAsImRvY3VtZW50X3R5cGUiOiJjaWUiLCJtcnRkIjp7ImRnMSI6IlVEeEp
WRUU4VTAxSlZFZzhQRXBQU0U0OFBFcFBTRTRnVTAxSlZFZzhQREU1T0RBME1UVThUVDxQTnpjM056SXpNUT09IiwiZGcxMS
I6Ik1USXpORFUyTnpnNVFVSkRSRVZHUjBoSlNrdE1UVTVQVUVGT1IxSlRWRlZXV0ZsYVUwRkVSVVU9Iiwic29kX21ydGQiO
iJNSUlGempDQ0JMYWdBd0lCQWdJSVFPWTJLSkdGVFVJd0RRWUpLb1pJaHZjTkFRRUxCUUF3WHpFTE1Baz0ifSwiaWFzIjp7
Imlhc19wayI6Ik1JSUJJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBejEyMzQ1Njc4OTA9Iiwic29
kX2lhcyI6Ik1JSUZhRENDQkZDZ0F3SUJBZ0lKQUwyS0pHRlRVSXdEUVlKS29aSWh2Y05BUUVMQlFBd1h6RUxNQT09IiwiY2
hhbGxlbmdlX3NpZ25lZCI6ImExYjJjM2Q0ZTVmNjc4OTAxMjM0NTY3ODkwMTIzNDU2Nzg5MGFiY2RlZjEyMzQ1Njc4OTBhY
mNkZWY9PSJ9fQ.xyz456abc789def012ghi345jkl678mno901pqr234stu567vwx890yz123",
  "mrtd_auth_session":"wxroVrBY2MCq4dDNGXACS",
  "mrtd_pop_nonce":"9f2c4a7e3b1d8c9a6e5f4b2a1c3d7e8f"
}
```

L'Istanza del Wallet DEVE:

- Eseguire lettura documento NFC conforme ICAO 9303 (PACE, ecc.).
- Validare firme crittografiche del documento e catene di certificati.
- Estrarre attributi di identità (DG1 e DG11), Anti-Cloning Public Key dai data groups del documento, e SODs (dalle Applicazioni MRTD e IAS).
- Eseguire l'Anti-Cloning Internal Authentication.

- Generare evidenza di validazione nel JWT.
- Autenticare utilizzando una Wallet Instance Attestation valida.
- Includere esatti `mrtd_auth_session` e `mrtd_pop_nonce` dalla risposta init.
- Firmare il `mrtd_validation_jwt` con la sua chiave privata.
- Gestire errori di lettura documento e fornire feedback appropriato.

Il Servizio MRTD PoP DEVE:

- Validare Wallet Instance Attestation secondo le specifiche IT-Wallet.
- Verificare firma OAuth-Client-Attestation-PoP.
- Validare che il parametro `mrtd_auth_session` corrisponda a una sessione attiva.
- Verificare che il `mrtd_pop_nonce` corrisponda al valore inviato nella risposta precedente.
- Analizzare e validare la firma (utilizzando la chiave pubblica dell'istanza del Wallet) e la struttura del `mrtd_validation_jwt`.
- Estrarre e verificare le evidence del documento dal `mrtd_validation_jwt`.
- Validare autenticità del documento utilizzando standard ICAO 9303.
- Verificare risposta challenge Anti-Cloning.
- Convalida delle prove crittografiche e delle catene di certificati del documento.
- Eseguire la correlazione dell'identità tra i dati del documento e il risultato dell'autenticazione LoA3.
- Controllare validità del documento (stato non di revoca).
- Verificare il binding tra le applicazioni IAS e MRTD controllando che il NUN estratto da DG1 sia presente (come valore hash) nell'IAS SOD, e che il DG1 stesso sia presente (come valore hash) nell'MRTD SOD. Questa doppia verifica assicura che entrambe le applicazioni risiedano sullo stesso chip fisico.

Risposta di Validazione MRTD PoP

Dopo completamento con successo di tutti i controlli da parte del Servizio MRTD PoP, DEVE inviare all'Istanza del Wallet una Risposta HTTP con codice *202 Accepted* includendo i seguenti parametri:

Tabella 12.12: Parametri Risposta di Validazione MRTD PoP

Parametro	Tipo	Descrizione
status	string	OBBLIGATORIO. DEVE essere valorizzato con *require_interaction*.
type	string	OBBLIGATORIO. DEVE essere valorizzato con *redirect_to_web*.
mrtd_val_pop_nonce	string	OBBLIGATORIO. Nonce di conferma finale per callback basato su browser.
redirect_uri	string	OBBLIGATORIO. URI di redirect del browser per il completamento del flusso autorizzativo.

Di seguito un esempio non normativo di una Risposta di Validazione MRTD PoP:

```
HTTP/1.1 202 Accepted
Content-Type: application/json; charset=utf-8

{
  "status": "require_interaction",
        "type": "redirect_to_web",
  "mrtd_val_pop_nonce": "0f2bff024317345b6927ce17e776361d",
  "redirect_uri":"https://eaa-provider.example.org/cb"
}
```

Il servizio MRTD PoP DEVE:

- Generare un nuovo nonce `mrtd_val_pop_nonce` per la conferma finale basata su browser.
- Restituire lo stato HTTP 202 per indicare il completamento dell'elaborazione asincrona.

L'istanza Wallet DEVE:

- Validare la response.
- Estrarre i parametri `mrtd_val_pop_nonce` e `redirect_uri` per preparare la successiva richiesta GET basata su browser.

Conferma Finale Basata su Browser

Dopo validazione MRTD PoP con successo, l'Istanza del Wallet DEVE reindirizzare l'User Agent al `challenge_redirect_uri` specificato nell'Authorization Details Object iniziale, includendo il `mrtd_val_pop_nonce` e `mrtd_auth_session` come parametri query:

```
https://eaa-provider.example.org/l2plus-callback?mrtd_val_pop_nonce=0f2bff024317345b6927ce17e7
76361d_signed&mrtd_auth_session=wxroVrBY2MCq4dDNGXACS
```

L'Istanza del Wallet DEVE validare che il `mrtd_val_pop_nonce` corrisponda al valore ricevuto dalla Risposta di Validazione MRTD PoP.

Il server di autorizzazione DEVE:

- Verificare che `mrtd_val_pop_nonce` corrisponda al valore inviato nella verification response e che sia firmato utilizzando la chiave privata dell'istanza Wallet.
- Verificare che il parametro `mrtd_auth_session` corrisponda a una sessione attiva.
- Verificare che tutti gli step di autenticazione (LoA3 + MRTD PoP) siano stati completati correttamente (inclusa la correlazione di identità recuperata tramite autenticazione LoA3 e quella presente nel documento).
- Generare l'authorization code finale.

Fase 4: Risposta di Autorizzazione OAuth

Dopo completamento con successo di tutti gli step di autenticazione e correlazione di identità, il Server di Autorizzazione DEVE emettere la Risposta di Autorizzazione OAuth finale. Se tutti i controlli di validazione sono stati superati, il Server di Autorizzazione DEVE reindirizzare l'User Agent nuovamente all'Istanza del Wallet con una Risposta di Autorizzazione OAuth includendo il codice di autorizzazione come definito negli step 6-7 di *Issuance Flow* (pagina 447), e nella Sezione credential-issuance-endpoint:Pushed Authorization Request (PAR) Response. Il Server di Autorizzazione DEVE utilizzare il `redirect_uri` incluso nel Request Object iniziale dall'Istanza del Wallet.

Gestione Errori

La gestione errori DEVE seguire le stesse regole come definite nella Sezione *Endpoint di Rilascio degli Attestati Elettronici* (pagina 595), riguardo ai formati e i riferimenti standard correlati.

Durante il flusso di Validazione MRTD PoP, quando si verificano errori recuperabili, il Servizio MRTD PoP PUÒ generare e restituire un nonce fresco per abilitare l'Utente a tentare nuovamente mantenendo sicurezza di sessione e prevenendo attacchi replay.

In aggiunta ai codici di errore già definiti nella Sezione *Endpoint di Rilascio degli Attestati Elettronici* (pagina 595), almeno i seguenti codici di errore DEVONO essere supportati.

Errori Risposta MRTD PoP

Tabella 12.13: Codici Errore Risposta MRTD PoP

Codice Errore	Descrizione
invalid_client	Autenticazione Istanza del Wallet fallita.
invalid_request	L'HTTP request non è valida o è malformata (struttura non corretta, dati mancanti, ecc.) oppure i parametri di sessione richiesti sono mancanti o non validi.
access_denied	L'utente non è idoneo per l'autenticazione eID Substantial con il meccanismo di verifica MRTD (ad esempio, CIE non trovata nel registro CIE)
temporarily_unavailable	Servizio di validazione documento o servizio Registro CIE è temporaneamente non disponibile.

Errori Risposta di Validazione MRTD PoP

Tabella 12.14: Codici Errore Risposta di Validazione MRTD PoP

Codice Errore	Descrizione
invalid_client	Autenticazione Istanza del Wallet fallita.
invalid_request	Richiesta di Validazione HTTP o JWT di Validazione è invalida o malformata (dovuto a struttura malformata, dati mancanti, fallimento firma, timeout richiesta, ecc.).
access_denied	Autenticazione utente o validazione documento fallita.
invalid_document	La convalida crittografica del documento non è riuscita (convalida SOD, verifica binding IAS/MRTD, stato di revoca, ecc.).
id_matching_failed	La corrispondenza tra l'identità ottenuta durante l'autenticazione primaria (eID LoA3) e quella ottenuta dalla PoP del Documento Elettronico non è riuscita.
temporarily_unavailable	Il servizio di validazione dei documenti o il servizio di registro CIE sono temporaneamente non disponibili.

Mappatura Codici di Stato HTTP

Le risposte di errore DEVONO utilizzare codici di stato HTTP appropriati:

- **400 Bad Request**: Per errori `invalid_request`.
- **401 Unauthorized**: Per errori `invalid_client`.
- **403 Forbidden**: Per errori `access_denied`.
- **422 Unprocessable Entity**: Per errori `invalid_document` o `id_matching_failed`.
- **503 Service Unavailable**: Per errori `temporarily_unavailable`.

Considerazioni di Sicurezza

Gestione Sicura di Sessione

Il parametro `mrtd_auth_session` serve come meccanismo di correlazione primario tra gli step di autenticazione. Le implementazioni DEVONO assicurare che questo identificatore abbia entropia sufficiente (minimo 128 bit) e sia crittograficamente sicuro. L'identificatore di sessione DEVE essere validato a ogni step per prevenire attacchi di session fixation.

Ogni step di autenticazione DEVE essere crittograficamente legato alla sessione OAuth tramite validazione JWT firmata per prevenire attacchi di session fixation, session confusion, e sostituzione di identità. Il Server di Autorizzazione DEVE mantenere la correlazione tra l'identità LoA3 e il proof del documento all'interno di un singolo contesto di sessione.

In particolare, ogni fase di autenticazione DEVE convalidare la correlazione tra:

- Coerenza dell'identificativo di sessione in tutte le fasi del protocollo.
- Attributi di identità LoA3 associati al contesto della sessione.
- Correlazione delle prove documentali con l'identità autenticata.
- Validazione della sequenza temporale per prevenire attacchi fuori ordine.

Quando i componenti operano al di fuori dei confini del Provider EAA, DEVONO essere implementate le seguenti misure di sicurezza aggiuntive:

- Comunicazione sicura tra servizi (ad esempio, tramite pinning del certificato, TLS reciproco, ecc.).
- Crittografia e integrità dei dati sensibili della sessione e/o delle informazioni di identità personale (ad esempio, utilizzando token JWE/JWS).
- Lock distribuiti per gli aggiornamenti dello stato di sessione

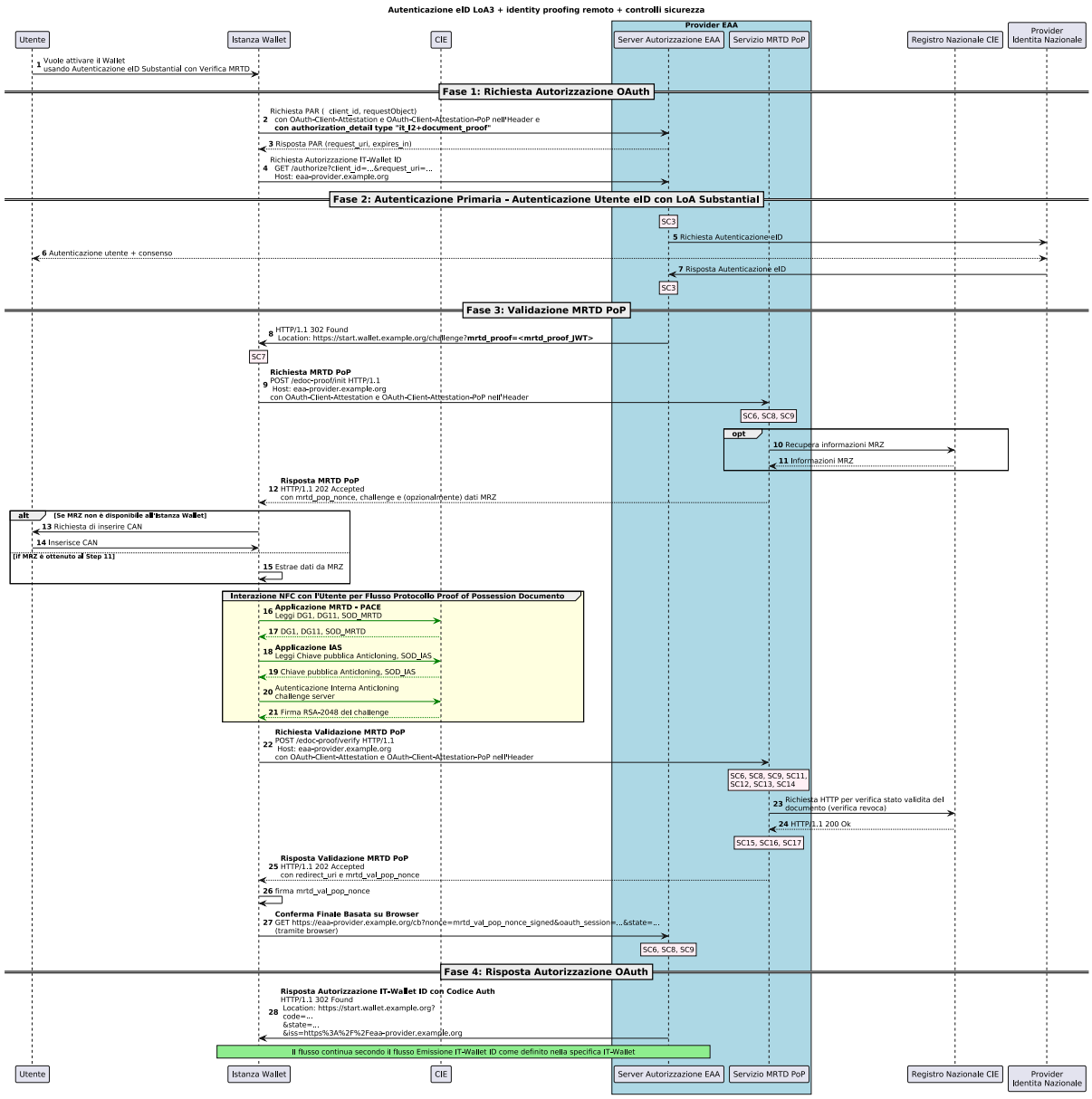


Fig. 12.11: Controlli di Sicurezza per Autenticazione eID Substantial con Verifica MRTD.

Generazione Challenge Crittografico

Il Servizio MRTD PoP DEVE generare challenge crittograficamente sicuri con entropia sufficiente (minimo 256 bits) per la validazione del documento. I valori di challenge DEVONO essere unici e NON DEVONO essere riutilizzati attraverso sessioni diverse (vincolata crittograficamente al contesto specifico della sessione OAuth) o tentativi di autenticazione. L'algoritmo di generazione challenge DOVREBBE incorporare l'identificatore `mrtd_auth_session` e timestamp per assicurare binding crittografico appropriato.

Gestione Ciclo di Vita Nonce

Ogni step nel flusso di Autenticazione eID Substantial con Verifica MRTD DEVE utilizzare valori nonce unici per prevenire attacchi replay. I valori nonce DEVONO avere tempi di scadenza appropriati e DEVONO essere invalidati dopo uso con successo. Il Server di Autorizzazione EAA e il Servizio MRTD PoP DEVONO mantenere validazione nonce sincronizzata per assicurare integrità di sessione.

Inoltre, ogni nonce ha uno scopo di sicurezza specifico: - `mrtd_pop_jwt_nonce` DEVE essere correlato con il JWT di prova MRTD. - `mrtd_pop_nonce` DEVE:

- Essere crittograficamente indipendente da `mrtd_pop_jwt_nonce`.
- Incorporare `mrtd_pop_jwt_nonce` come input per mantenere la catena di fiducia.
- Utilizzare una diversa sorgente di entropia per prevenire attacchi di correlazione.
- `mrtd_val_pop_nonce` DEVE:
- Essere firmato dalla chiave privata dell'istanza del wallet.
- Includere la convalida del timestamp anti-replay.
- Essere verificato rispetto all'intera catena di nonce per verificarne l'integrità.

Metadati del Provider EAA

In aggiunta ai valori `trust_frameworks_supported` definiti nella sezione credential-issuer-metadata:Metadata per openid_credential_issuer, i Metadati del Provider EAA per `openid_credential_issuer` DEVONO anche supportare il valore `it_l2+document_proof` indicante il protocollo di Autenticazione multi-step descritto in questa Specifica.

Controlli di Sicurezza

I seguenti controlli di sicurezza DEVONO essere implementati nel protocollo:

Tabella 12.15: Controlli di Sicurezza per Autenticazione eID Substantial con Verifica MRTD

#	Descrizione	Fase
SC1	I canali di comunicazione di rete sono protetti utilizzando TLS v1.2+ con cifrari sicuri.	Tutte
SC2	L'Istanza del Wallet assicura l'autenticità del Provider EAA e del Provider di Identità eID tramite pinning del certificato leaf di ogni server.	Tutte
SC3	Il Server di Autorizzazione EAA verifica che il Provider di Identità eID utilizzato nella fase *eID LoA3* sia accreditato.	Fase 2
SC4	Il Server di Autorizzazione EAA verifica l'autenticità e integrità dei dati estratti dall'asserzione *eID LoA3*, controllando la firma digitale.	Fase 2
SC5	Il valore challenge e tutti i valori nonce sono generati con protezioni per prevenire attacchi bruteforce o di indovinamento.	Fase 3
SC6	Il Provider EAA verifica tutti i valori nonce per rilevare attacchi replay.	Fase 3
SC7	L'Istanza del Wallet verifica che `challenge_info` sia firmato appropriatamente dal Server di Autorizzazione EAA. Inoltre, controlla che `challenge_info` contenga: un valore `iss` corrispondente al valore del Server di Autorizzazione EAA; un valore aud uguale al `client_id` dell'Istanza del Wallet; e un valore `state` uguale a quello nella richiesta PAR, per essere sicuri che la risposta sia legata alla richiesta iniziale fatta dall'Istanza del Wallet nello Step 2. Quindi le informazioni fornite come parte di `challenge_info`, in particolare l'`htu` che corrisponde all'url di redirect da seguire per il *MRTD PoP*, non sono manomesse.	Fase 3
SC8	Il Provider EAA controlla che l'`mrtd_auth_session` sia associata alla stessa Istanza del Wallet in tutte le richieste nella fase *MRTD PoP*. Quindi il Provider EAA può essere sicuro che l'Istanza del Wallet che esegue la fase *MRTD PoP*: sia fidata; sia sempre la stessa attraverso il protocollo; e abbia precedentemente iniziato l'emissione IT-Wallet ID (richiesta PAR). Questo può essere implementato richiedendo all'Istanza del Wallet di eseguire un proof of possession della sua chiave privata (es., all'interno di OAuth-Client-Attestation o firmando un valore nonce).	Fase 3
SC9	Il Provider EAA controlla che l'`mrtd_auth_session` non sia scaduta (timeout di validità tipicamente di 5 minuti), cioè che l'operazione sia stata conclusa entro una certa quantità di tempo.	Fase 3
SC10	L'integrità e riservatezza del canale tra la *CIE fisica* e il *device_wallet fisico* è protetta con il protocollo PACE (tramite l'algoritmo e le funzioni di derivazione chiave supportate dalla carta).	Fase 3
SC11	Il Servizio MRTD PoP verifica l'autenticità e integrità del `mrtd_validation_jwt` controllando che sia firmato con la chiave privata dell'Istanza del Wallet associata all'`mrtd_auth_session`.	Fase 3

Requisiti implementativi aggiuntivi:

- **Rate limiting**: Protezione contro attacchi automatizzati e tentativi bruteforce.
- **Session timeout**: Pulizia automatica di sessioni di autenticazione incomplete.
- **Audit logging**: Logging completo di tutti gli eventi di autenticazione con identificatori di correlazione.
- **Error handling**: Risposte di errore sicure che non fanno trapelare informazioni sensibili.
- **Cryptographic material cleanup**: Eliminazione sicura di chiavi temporanee e challenge.

Considerazioni Implementative

Le implementazioni DEVONO incorporare meccanismi di rate-limiting per proteggere contro attacchi automatizzati ed esaurimento risorse. Le implementazioni DOVREBBERO configurare un timeout di interazione che bilanci esperienza utente e postura di sicurezza, accomodando la variabilità inerente nella lettura di documenti basata su NFC.

Il Provider EAA DEVE implementare timeout di sessione con meccanismi di pulizia appropriati, assicurando che le risorse di sessione siano rilasciate e il materiale crittografico temporaneo sia eliminato in modo sicuro quando le sessioni scadono.

Tutti gli eventi rilevanti per la sicurezza durante il flusso di Autenticazione eID Substantial con Verifica MRTD DEVONO essere loggati con dettaglio sufficiente per scopi di auditing preservando la privacy dell'Utente, assicurando che le informazioni di identificazione personale, quando memorizzate, siano hashate appropriatamente. I log di audit DEVONO avere identificatori di correlazione consistenti, abilitando tracciamento end-to-end attraverso tutte le fasi del protocollo, con protezione di integrità crittografica per prevenire manomissioni.

12.2 Presentazione dell'Attestato Elettronico

Questa sezione descrive come un'Istanza di Relying Party richiede a un'Istanza del Wallet la presentazione dell'Attestato Elettronico di Dati di Identificazione Personale/Attestati Elettronici di Attributi.

In questa sezione vengono descritti i seguenti flussi:

- *Flusso Remoto* (pagina 509), dove l'Utente presenta un Attestato Elettronico a un'Istanza di Relying Party web secondo OpenID4VP. In questo scenario, l'user-agent e l'Istanza del Wallet possono essere utilizzati nello stesso dispositivo (**Same Device Flow**), o in dispositivi diversi (**Cross Device Flow**).
- *Flusso di Prossimità* (pagina 538), dove l'Utente presenta un Attestato Elettronico a un'Istanza di Relying Party mobile secondo ISO18013-5. L'Utente interagisce con un

Verificatore di Attestati Elettronici utilizzando tecnologie di connessione di prossimità come i Codici QR e il Bluetooth Low Energy (BLE).

- *Identity Matching e Identity Reconciliation* (pagina 567), applicato dopo la verifica crittografica di un Attestato Elettronico presentato.

La presentazione in prossimità o comunque offline di un Attestato Elettronico DEVE riuscire solo quando le chiavi private vincolate sono memorizzate in un Keystore locale o in un WSCD locale. Gli Attestati Elettronici le cui chiavi private vincolate sono memorizzate in un WSCD remoto NON DEVONO essere presentati offline (*WP_160* (pagina 781)).

Nota

Quando una Relying Party deve distinguere l'Utente in fase di autenticazione, DEVE farlo sulla base delle informazioni contenute negli Attestati Elettronici richiesti nella richiesta di presentazione. I claim corrispondenti devono essere inclusi nella richiesta di presentazione. La Relying Party distingue l'Utente valutando i valori dei claim effettivamente presentati.

Di seguito sono riportati esempi non normativi.

1. **PID o IT-Wallet ID.** La Relying Party richiede i claim di identificazione previsti dal relativo modello dati, ad esempio `given_name`, `family_name` e `personal_administrative_number` per il PID, oppure `tax_id_code` per l'IT-Wallet ID. Vedere *Modello di Dati del PID* (pagina 404) e *Modello di Dati dell'IT-Wallet ID* (pagina 416).

   Esempio di `dcql_query` nel flusso remoto (PID in formato SD-JWT VC):

   ```
   {
     "credentials": [
       {
         "id": "pid",
         "format": "dc+sd-jwt",
         "meta": {
           "vct_values": [ "urn:eudi:pid:it:1" ]
         },
         "claims": [
           {"path": ["given_name"]},
           {"path": ["family_name"]},
           {"path": ["personal_administrative_number"]}
         ]
       }
     ]
   }
   ```

   Gli stessi claim, nel flusso di prossimità, sono richiesti nell'`ItemsRequest` mdoc:

   ```
   {
     "docType": "eu.europa.ec.eudi.pid.1",
   ```

```
    "nameSpaces": {
      "eu.europa.ec.eudi.pid.1": {
        "given_name": false,
        "family_name": false,
        "personal_administrative_number": false
      }
    }
}
```

2. **Altro Attestato Elettronico di catalogo.** La Relying Party richiede i claim previsti dallo schema dell'Attestato Elettronico pubblicato nel *Catalogo degli Attestati Elettronici* (pagina 190). Il relativo modello dati è definito dallo schema corrispondente nel *Registro degli Schema* (pagina 232).

   Esempio di `dcql_query` nel flusso remoto per una mDL (`credential_type` `mDL`, formato mdoc-CBOR, `docType` `org.iso.18013.5.1.mDL`):

```
{
  "credentials": [
    {
      "id": "mobile driving license",
      "format": "mso_mdoc",
      "meta": {
        "doctype_value": "org.iso.18013.5.1.mDL"
      },
      "claims": [
        {"path": ["org.iso.18013.5.1", "given_name"]},
        {"path": ["org.iso.18013.5.1", "family_name"]},
        {"path": ["org.iso.18013.5.1", "document_number"]}
      ]
    }
  ]
}
```

Nota

In caso di utilizzo di credenziali ottenute in batch, l'Istanza del Wallet DOVREBBE implementare una logica di selezione delle Credenziali (ad esempio, basata sulla prima in scadenza) e DEVE contrassegnarla come consumata. Al termine del flusso, DEVE ridurre il numero di credenziali disponibili nel batch e, in base a ciò, può verificare se è il momento di richiedere un nuovo batch di credenziali.

12.2.1 Flusso Remoto

A seconda di come l'Utente stia interagendo con il frontend dell'App di Verifica Web, usando cioè il dispositivo in cui risiede l'Unità Wallet (**Same Device**) oppure un altro dispositivo (**Cross Device**), la Relying Party DEVE supportare i seguenti flussi remoti (RPR-84):

- **Same Device**: essa DEVE fornire un indirizzo `HTTP` all'Istanza del Wallet utilizzando un *redirect* (`302`) o un href HTML nella pagina web (RPR-01);
- **Cross Device**: essa DEVE fornire un indirizzo `HTTP` tramite un Codice QR che l'Utente scansiona con la fotocamera del proprio dispositivo oppure con l'Istanza del Wallet (RPR-03).

Per richiamare l'Istanza del Wallet corretta, la Relying Party DOVREBBE invocare l'Istanza del Wallet installata sul dispositivo dell'Utente che l'Utente desidera utilizzare. Queste informazioni DOVREBBERO essere fornite dall'Utente utilizzando la Selection Page descritta in *Design dell'Esperienza Utente* (pagina 39).

- Se la Selection Page è disponibile, l'utente seleziona l'Istanza del Wallet, quindi la Relying Party recupera i Wallet metadata come descritto in *Flusso di Recupero dei Wallet Metadata* (pagina 570). Il contenuto del HTML href o del QR Code dipende dal parametro `authorization_endpoint` nei Wallet metadata:
- Se `authorization_endpoint` è disponibile e contiene un URL HTTPS (Universal Link), la Relying Party DOVREBBE utilizzare quell'endpoint.
- Altrimenti, la Relying Party DEVE utilizzare uno degli schemi URL personalizzati: `openid4vp://` (come definito nella Sezione 13.1.2 di [OpenID4VP]) o `haip-vp://` (come definito nella Sezione 5.1 di [OPENID4VC-HAIP]). L'Istanza del Wallet DEVE supportare entrambi gli schemi URL personalizzati.
- Nel caso in cui la Relying Party non supporti la Selection Page o il recupero dei Wallet metadata fallisce per qualche motivo, la Relying Party richiamerà l'Istanza del Wallet utilizzando uno degli schemi URL personalizzati descritti precedentemente.

Successivamente all'invocazione dell'Istanza del Wallet, l'Istanza del Wallet valida la trust con la Relying Party e ne valuta la richiesta. Se valida e l'Utente fornisce il consenso per la divulgazione dei propri Attestati Elettronici, li invia sotto forma di Verifiable Presentation.

Una descrizione ad alto livello del flusso remoto, dal punto di vista dell'Utente, è fornita di seguito e mostrata in *Flusso di Protocollo Remoto di Alto Livello.* (pagina 510):

1. *Authorization Request*: l'Istanza del Wallet ottiene un `URL` nel flusso Same Device o un `QR Code` nel flusso Cross Device contenente il Request Object firmato:
    - direttamente, passando un Request Object by value (tramite il parametro `request`), oppure

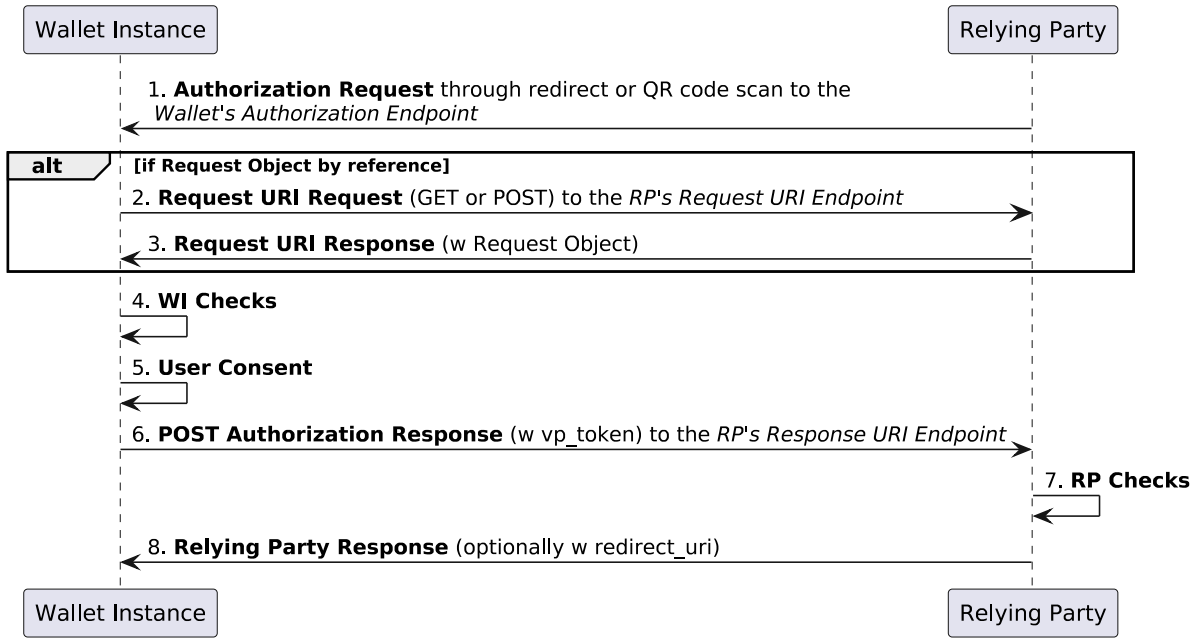


Fig. 12.12: Flusso di Protocollo Remoto di Alto Livello.

- passando un Request Object by reference (tramite il parametro `request_uri`), dove il Request Object firmato è disponibile per il download. Questo è l'unico metodo consentito per le Relying Party conformi all'OpenID4VC High Assurance Interoperability Profile (HAIP) OPENID4VC-HAIP.

Se il Request Object è passato by reference, vengono eseguiti anche i Passaggi 2 e 3.

2. *Richiesta URI Request*: l'Istanza del Wallet estrae dal payload i seguenti parametri: `client_id`, `request_uri`, `request_uri_method`.

- Se `request_uri_method` è fornito e impostato con il valore `post`, l'Istanza del Wallet DOVREBBE trasmettere i suoi metadata all'endpoint `request_uri` della Relying Party utilizzando il metodo HTTP `POST`.

- Se `request_uri_method` è impostato con il valore `get` o non è presente, l'Istanza del Wallet DEVE recuperare il Request Object firmato utilizzando una richiesta HTTP con metodo `GET` all'endpoint fornito nel parametro `request_uri` (RPR-08).

3. *URI Request Response*: la Relying Party restituisce un Request Object firmato all'Istanza del Wallet.

4. *Controlli Istanza di Wallet*: l'Istanza del Wallet:

a. verifica la firma del Request Object firmato utilizzando la chiave pubblica identificata nell'intestazione JWT del Request Object. Utilizzando tale

riferimento, l'Istanza del Wallet è in grado di selezionare la corretta chiave pubblica della Relying Party per la verifica della firma (*WP_085* (pagina 798)).

b. verifica che il `client_id` contenuto nel Request Object (Relying Party) corrisponda a quello ottenuto al passaggio 2:

   - Se `client_id` utilizza il prefisso `openid_federation`, DEVE corrispondere al parametro `sub` contenuto nella Entity Configuration della Relying Party all'interno della Trust Chain (*WP_086* (pagina 798)).
   - Se `client_id` utilizza il prefisso `x509_hash`, l'Istanza del Wallet DEVE verificare che l'hash del certificato X.509 della Relying Party (nell'intestazione `x5c` della richiesta) corrisponda all'hash contenuto in `client_id` del passaggio 2 (come definito in OpenID4VP, Sezione 5.9.3).

c. valuta gli Attestati Elettronici richiesti e verifica l'idoneità della Relying Party nel richiedere questi ultimi. Ad esempio, applicando le politiche relative a quella specifica Relying Party ottenute con la Trust Chain (*WP_087* (pagina 798)).

5. *Risposta di Autorizzazione POST*: l'Istanza del Wallet presenta le informazioni richieste alla Relying Party.

6. *Controlli RP*: La Relying Party convalida le Credenziali presentate verificando la fiducia con i loro Fornitori di Attestati Elettronici e controlla i rispettivi stati di validità.

7. *Risposta della Relying Party*: l'Istanza del Wallet informa l'Utente dell'autenticazione riuscita con la Relying Party, e l'Utente continua la navigazione.

Di seguito è riportato un diagramma di sequenza che dettaglia le interazioni tra tutte le parti coinvolte.

I dettagli di ogni passaggio mostrato nell'immagine precedente sono descritti di seguito.

Passaggi 1-2: L'Utente richiede di accedere a una risorsa protetta della Relying Party.

Passaggio 3: La Relying Party crea un valore *state* fresco, crittograficamente casuale e con entropia sufficiente, lo associa alla sessione dello user-agent (ad esempio, utilizzando un cookie HTTP protetto) e lo memorizza lato server con un tempo di scadenza breve. Quindi ispeziona lo user-agent per determinare se il flusso avviene sullo stesso dispositivo dello user-agent.

Passaggi 4-7 (Authorization Request): La Relying Party fornisce allo user-agent una pagina JavaScript che ispeziona lo *state endpoint* e all'Istanza del Wallet un URL contenente l'Authorization Request.

Nel **Flusso Cross Device**, l'URI dell'Authorization Request viene presentato attraverso un Codice QR mostrato all'Utente. L'Utente scansiona il Codice QR

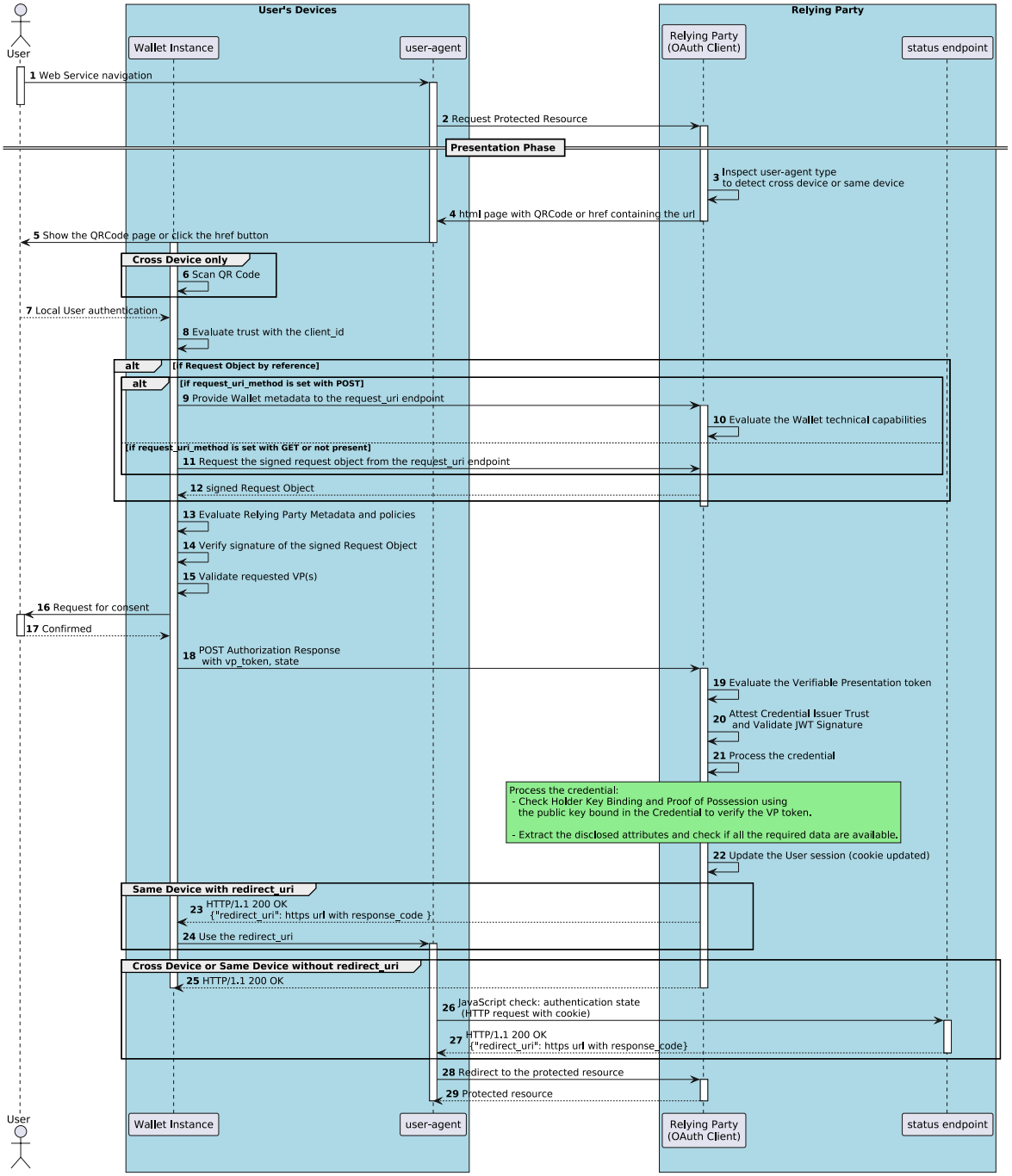


Fig. 12.13: Flusso di Protocollo Remoto.

utilizzando l'Istanza del Wallet e recupera un URL. Di seguito è rappresentato un esempio non normativo di un Codice QR emesso dalla Relying Party.



Nota

Il *livello di correzione degli errori* scelto per il Codice QR DEVE essere Q (Quartile - fino al 25%), poiché offre un buon equilibrio tra capacità di correzione degli errori e densità/spazio dei dati. Questo livello di qualità e correzione degli errori consente al Codice QR di rimanere leggibile anche se è danneggiato o parzialmente oscurato (RPR-77).

Se il Request Object viene passato per valore (by value), l'URL all'interno del codice QR contiene i parametri `client_id` e `request`. Di seguito è rappresentato un esempio non normativo del contenuto del Codice QR per un Request Object by value:

```
https://wallet-solution.example.org/authorization?client_id=openid_federation%3A%2F%2Frel␣
ying-party.example.org&request=eyJhbGciOiJFUzI1NiIs...9t2LQ
```

Se invece il Request Object viene passato per riferimento (by reference), l'URL all'interno del codice QR contiene i parametri `client_id`, `request_uri` e `request_uri_method` (*WP_076–077* (pagina 798)). Di seguito è riportato un esempio non normativo del contenuto del codice QR per un Request Object by reference:

```
https://wallet-solution.example.org/authorization?client_id=openid_federation%3A%2F%2Frel␣
ying-party.example.org&request_uri=https%3A%2F%2Frelying-party.example.org&request_uri_me␣
thod=post
```

Un template HTML ufficiale e autoconsistente per questa pagina QR **Cross Device**—con header, footer, accessibilità, testi multilingua e payload dimostrativo

configurabile—è disponibile nella sezione official-resources:Componenti HTML (**IT-Wallet Presentation QR Code Page**). È raggiungibile dalle card della **IT-Wallet Selection Page** nella stessa sezione.

Mentre, nel **Flusso Same Device**, la Relying Party risponde tramite HTTP Response Redirect (con codice di stato impostato a `302`) o mostra all'utente una pagina html con un pulsante href, aventi l'URL che fornisce le stesse informazioni del Flusso Cross Device (*WP_076–077* (pagina 798)). Di seguito è riportato un esempio non normativo per un Request Object by reference:

```
HTTP/1.1 302 Found
Location: https://wallet-solution.digital-strategy.europa.eu?client_id=openid_federation%3
A%2F%2Frelying-party.example.org%2Fcb&request_uri=https%3A%2F%2Frelying-party.example.org%
2Frequest_uri&request_uri_method=post
```

Passaggio 8: L'Istanza del Wallet valuta la trust con la Relying Party (*WP_078–080* (pagina 798)).

Passaggi 9-11 (Richiesta URI Request): L'Istanza del Wallet verifica se la Relying Party ha fornito il request_uri_method all'interno del suo Request Object firmato (*WP_083* (pagina 798)).

- Se è fornito ed è uguale a post, l'Istanza del Wallet DOVREBBE fornire i suoi metadata alla Relying Party. La Relying Party aggiorna il Request Object in base alle capacità tecniche del Wallet.

  Di seguito è riportato un esempio non normativo di una richiesta HTTP effettuata dall'Istanza del Wallet alla Relying Party.

```
POST /request HTTP/1.1
Host: client.example.org
Content-Type: application/x-www-form-urlencoded
Accept: application/oauth-authz-req+jwt

wallet_metadata=%7B%22authorization_endpoint%22%3A%20%22https%3A%2F%2Fwallet-solution.digi
tal-strategy.europa.eu%2Fauthorization%22%2C%20%22response_types_supported%22%3A%20%5B%22v
p_token%22%5D%2C%20%22vp_formats_supported%22%3A%20%7B%22dc%2Bsd-jwt%22%3A%20%7B%22sd-jwt_
alg_values%22%3A%20%5B%22ES256%22%2C%20%22ES384%22%5D%7D%2C%22mso_mdoc%22%3A%7B%22issuerau
th_alg_values%22%3A%5B-9%2C-51%5D%2C%22deviceauth_alg_values%22%3A%5B-9%2C-51%5D%7D%7D%2C%
22request_object_signing_alg_values_supported%22%3A%20%5B%22ES256%22%5D%2C%22client_id_pre
fixes_supported%22%3A%5B%22openid_federation%22%2C%22x509_hash%22%5D%7D%26wallet_nonce%3Dq
PmxiNFCR3QTm19POc8u
```

  Dove il corpo della richiesta prima di essere codificato in `application/x-www-form-urlencoded` dal Wallet corrisponde a:

```
{
  "wallet_metadata": {
```

```
    "authorization_endpoint": "https://wallet-solution.digital-strategy.europa.eu/authoriz
ation",
    "response_types_supported": [
      "vp_token"
    ],
    "vp_formats_supported": {
      "dc+sd-jwt": {
        "sd-jwt_alg_values": ["ES256","ES384"]
      },
      "mso_mdoc": {
        "issuerauth_alg_values": [-9,-51],
        "deviceauth_alg_values": [-9,-51]
      }
    },
    "request_object_signing_alg_values_supported": ["ES256"],
    "client_id_prefixes_supported": ["openid_federation","x509_hash"]
  },
  "wallet_nonce": "qPmxiNFCR3QTm19POc8u"
}
```


- Quando la Relying Party non supporta `request_uri_method` con valore `post`, l'Istanza del Wallet richiede il Request Object firmato utilizzando il metodo HTTP GET (*WP_082* (pagina 798)).

Passaggio 12 (URI Request Response): La Relying Party emette il Request Object firmandolo utilizzando una delle sue chiavi crittografiche private, le cui corrispondenti chiavi pubbliche sono state pubblicate all'interno della sua Entity Configuration (*metadata.openid_credential_verifier.jwks*) come estratto dall'Istanza del Wallet per *WP_084* (pagina 798). L'Istanza del Wallet ottiene il Request Object firmato.

Di seguito è riportato un esempio non normativo della Risposta URI di Reindirizzamento:


```
HTTP/1.1 200 OK
Content-Type: application/oauth-authz-req+jwt

eyJhbGciOiJFUzI1NiIs...9t2LQ
```


Passaggi 13-15 (Controlli WI): L'Istanza del Wallet verifica l'Oggetto di Richiesta, che è sotto forma di JWT firmato (*WP_085–086* (pagina 798)).

Un esempio non normativo di un Oggetto di Richiesta sotto forma di intestazione e payload decodificati è mostrato di seguito:


```
{
  "alg": "ES256",
  "typ": "oauth-authz-req+jwt",
  "kid": "9tjiCaivhWLVUJ3AxwGGz_9",
```

```
  "trust_chain": [
    "MIICajCCAdOgAwIBAgIC...awz",
    "MIICajCCAdOgAwIBAgIC...2w3",
    "MIICajCCAdOgAwIBAgIC...sf2"
  ],
  "x5c": [
    "MIIDqjCCApKgAwIBAgIESLNEvDA ...",
    "MIICwzCCAasCCQCKVy9eKjvi+jA ...",
    "MIIDTDCCAjSgAwIBAgIJAPlnQYH..."
  ]
}
```

```
{
  "client_id": "openid_federation:https://relying-party.example.org",
  "response_mode": "direct_post.jwt",
  "response_type": "vp_token",
  "dcql_query": {
    "credentials": [
      {
        "id": "personal id data",
        "format": "dc+sd-jwt",
        "meta": {
          "vct_values": [ "urn:eudi:pid:it:1" ]
        },
        "claims": [
          {"path": ["given_name"]},
          {"path": ["family_name"]},
          {"path": ["birthdate"]}
        ]
         },
        {
        "id": "mobile driving license",
        "format": "mso_mdoc",
        "meta": {
          "doctype_value": "org.iso.18013.5.1.mDL"
        },
        "claims": [
          {"path": ["org.iso.18013.5.1", "given_name"]},
          {"path": ["org.iso.18013.5.1", "family_name"]},
          {"path": ["org.iso.18013.5.1", "document_number"]}
        ]
      }
    ]
  },
  "response_uri": "https://relying-party.example.org/response_uri",
  "nonce": "2c128e4d-fc91-4cd3-86b8-18bdea0988cb",
  "wallet_nonce": "qPmxiNFCR3QTm19POc8u",
```

```
  "client_metadata": {
    "jwks": {
      "keys": [
        {
          "kty": "EC",
          "use": "enc",
          "crv": "P-256",
          "x": "f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
          "y": "x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
          "kid": "20260202-abc123",
          "alg": "ECDH-ES"
        }
      ]
    },
    "encrypted_response_alg_values_supported": ["ECDH-ES"],
    "encrypted_response_enc_values_supported": ["A128GCM", "A256GCM"]
  },
  "state": "3be39b69-6ac1-41aa-921b-3e6c07ddcb03",
  "iss": "https://relying-party.example.org",
  "iat": 1672418465,
  "exp": 1672422065
}
```


Quindi elabora i metadati della Relying Party e applica le politiche pertinenti per determinare quali Credenziali Elettroniche e dati dell'Utente la Relying Party è autorizzata a richiedere (*WP_087* (pagina 798)).

Passaggi 16-17 (Consenso dell'Utente): L'Istanza del Wallet richiede il consenso dell'Utente per divulgare gli Attetstati Elettronici richiesti mostrando l'identità della Relying Party e gli attributi richiesti. L'Utente autorizza e acconsente alla presentazione degli Attributi Elettronici selezionando e o deselezionando i dati personali da rilasciare (*WP_088* (pagina 798)).

Passaggio 18 (Authorization Response): L'Istanza del Wallet fornisce la Authorization Response alla Relying Party utilizzando una richiesta HTTP con il metodo POST utilizzando modalità di risposta `direct_post.jwt`.

Di seguito è riportato un esempio non normativo della Authorization Response:


```
POST /response_uri HTTP/1.1
HOST: relying-party.example.org
Content-Type: application/x-www-form-urlencoded

response=eyJhbGciOiJFQ0RILUVTIiwiZW5jIjoiQTI1NkdDTSIsImtpZCI6ImVwaGVtZXJhbC0yMDI2MDIwMi1hY
mMxMjMiLCJlcGsiOnsi...fX0..5vL9d2X8fQ..dGhpcy1pcy1hLXNhbXBsZS1jaXBoZXJ0ZXh0.ABCDEFGHIJKLMN
OPQRS
```


Di seguito è riportato un esempio non normativo che mostra l'intestazione protetta del JWE decifrata e il payload del JWT contenuto nel parametro `response`, prima

della codifica base64url. Il valore del parametro `vp_token` corrisponde al formato utilizzato quando il linguaggio di query DCQL è utilizzato nella richiesta di presentazione.


```
{
  "alg": "ECDH-ES",
  "enc": "A256GCM",
  "kid": "20260202-abc123",
  "epk": {
    "kty": "EC",
    "crv": "P-256",
    "x": "f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
    "y": "x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0"
  }
}
```

```
{
  "state": "3be39b69-6ac1-41aa-921b-3e6c07ddcb03",
  "vp_token": {
    "personal id data": ["eyJhbGciOiJFUzI1NiIs...PT0iXX0"],
    "mobile driving license": ["o2Nkb2N0eXBlb3Jzby4xO...Nib3JfZHVtbXk"]
  }
}
```


Nota

Quando restituisce una Credenziale richiesta nel formato `mso_mdoc` all'interno di `vp_token`, il Wallet DEVE vincolare crittograficamente la presentazione mdoc risultante alla corrente transazione OpenID4VP. A tal fine, il Wallet costruisce il `SessionTranscript` ISO utilizzato per l'autenticazione del dispositivo mdoc e applica le regole di profilazione di OpenID4VP impostando `DeviceEngagementBytes` a `null` e `EReaderKeyBytes` a `null`, e impostando il relativo campo `Handover` a una struttura definita da OpenID4VP (`OpenID4VPHandover`) derivata dai parametri della Authorization Request. Il Wallet calcola quindi l'autenticazione del dispositivo mdoc (firma del dispositivo) su dati che includono tale `SessionTranscript`, in modo che la presentazione mdoc risultante sia valida solo per quella specifica transazione OpenID4VP. Per la definizione normativa di `OpenID4VPHandover` e delle corrispondenti regole di profilazione del `SessionTranscript`, vedere OpenID4VP Appendice B.2.

Passaggi 19-22 (Controlli RP): La Relying Party verifica la Risposta di Autorizzazione, estrae la Wallet Attestation per stabilire la fiducia con la Soluzione Wallet. Quindi estrae il `vp_token` che contiene una o più presentazioni di Attestati Elettronici di Attributi, e ne valida il formato complessivo. Per ogni presentazione di un Attestato, la Relying Party ne verifica l'integrità secondo i criteri della query DCQL definiti nella Richiesta di Autorizzazione. La Relying Party DEVE anche attestare la fiducia con il relativo Fornitore di Attestati Elettronici

e verificare la prova di possesso dell'Istanza del Wallet per ogni Attestato presentato. Infine, la Relying Party verifica lo stato di revoca di ogni Attestato presentato come descritto in *Revoca e Sospensione degli Attestati Elettronici* (pagina 426). Se tutte le verifiche precedenti hanno dato esito positivo, la Relying Party aggiorna la sessione dell'Utente.

Passaggi 23-24 o 25 (Risposta della Relying Party): La Relying Party fornisce all'Istanza del Wallet la risposta sulla presentazione, che a sua volta informa l'Utente.

> Dopo aver ricevuto e convalidato la Authorization Response al Response Endpoint, la Relying Party restituisce all'Istanza del Wallet un codice di stato HTTP 200 OK. In particolare, nel Flusso Same Device, la Relying Party DEVE anche passare il parametro `redirect_uri` nella risposta all'Istanza del Wallet. Dopo aver ricevuto il `redirect_uri`, l'Istanza del Wallet DEVE eseguire un reindirizzamento all'URL specificato dal `redirect_uri`. Questo reindirizzamento consente alla Relying Party di riprendere senza problemi l'interazione con l'Utente sul dispositivo che ha avviato il flusso. Quando la risposta non contiene il parametro `redirect_uri`, l'Istanza del Wallet non è tenuta a eseguire ulteriori passaggi. L'Utente dovrebbe chiudere manualmente l'Istanza del Wallet e aprire lo user-agent per continuare il flusso (RPR-83).
>
> Di seguito è riportato un esempio non normativo della risposta nel Flusso Same Device.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "redirect_uri": "https://relying-party.example.org/cb?response_code=091535f699ea575c7937fa5f0
f454aee"
}
```

Passaggi 26-27: La pagina JavaScript continua ad ispezionare lo Status Endpoint.

> Di seguito è riportato un esempio non normativo della Richiesta HTTP allo Status Endpoint, dove il parametro `id` contiene un valore opaco e casuale:
>
> ```
> GET /session-state?id=3be39b69-6ac1-41aa-921b-3e6c07ddcb03 HTTP/1.1
> HOST: relying-party.example.org
> ```
>
> Quando l'Istanza del Wallet ha fornito la presentazione all'endpoint **response_uri** della Relying Partye, nel Flusso Same Device, lo user-agent è ritornato correttamente tramite `redirect_uri` all'interno della stessa sessione utente, e l'autenticazione dell'Utente ha avuto successo. La Relying Party aggiorna il cookie di sessione consentendo all'user-agent di accedere alla risorsa protetta. Viene fornito un URL di reindirizzamento che trasporta la posizione in cui l'user-agent deve navigare. Di seguito è riportato un esempio non normativo della risposta con il redirect_uri dalla Relying Party all'user-agent.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "redirect_uri": "https://relying-party.example.org/cb?response_code=091535f699ea575c7937
fa5f0f454aee"
}
```

Passaggi 28-29: Lo user-agent viene reindirizzato al `redirect_uri` per continuare la navigazione con la risorsa protetta resa disponibile all'Utente (*WP_094* (pagina 798)). La Relying Party DEVE considerare la transazione completata solo se il reindirizzamento di ritorno viene ricevuto nella stessa sessione utente in cui il flusso è stato avviato; in caso contrario, DEVE rifiutare la presentazione.

Nota

Durante ciascuna transazione di presentazione di credenziali eseguita tramite il flusso remoto, l'Istanza del Wallet DEVE creare e mantenere un corrispondente record di transazione nel registro delle transazioni (vedere *Dashboard dell'Istanza del Wallet e Registrazione delle Transazioni* (pagina 319)).

Il record di transazione DEVE essere creato una volta che l'Istanza del Wallet ha accettato la richiesta di presentazione per l'elaborazione (ossia, dopo la validazione della richiesta e i controlli di trust/policy della Relying Party, Passaggi 13–15). A questo punto, il record DEVE includere i metadati della transazione e il contesto della richiesta disponibile in quella fase (ad esempio, il/i tipo/i di Credenziale richiesti e l'/gli identificativo/i degli attributi richiesti), senza registrare alcun valore degli attributi.

Il record DEVE essere aggiornato man mano che la transazione procede, in modo da riflettere l'evoluzione dello stato della transazione e il contesto del risultato (ad esempio, quanto effettivamente presentato dopo il consenso dell'Utente e la preparazione/invio della risposta, Passaggi 16–18), senza registrare alcun valore degli attributi.

Il record DEVE essere finalizzato al termine della transazione, indicando l'esito (ad esempio, completata, fallita o interrotta; Passaggi 23–29).

Authorization Request

I parametri URL contenuti nella Authorization Request della Relying Party sono descritti nella tabella seguente.

Nome	Descrizione
client_id	OBBLIGATORIO. Identificatore univoco della Relying Party. Il valore DEVE utilizzare uno dei seguenti prefissi di identificatore del Client (come definito in OpenID4VP, Sezione 5.9): `openid_federation` (identificatore dell'entità della Relying Party in una catena di fiducia) oppure `x509_hash` (hash SHA-256 codificato in base64url del certificato X.509 della Relying Party).
request	CONDIZIONALE. OBBLIGATORIO a meno che `request_uri` non sia presente. Contiene il Request Object firmato e codificato in base64url. Per il contenuto dell'oggetto Request, vedere la Sezione *Request Object* (pagina 525).
request_uri	CONDIZIONALE. OBBLIGATORIO a meno che `request` non sia presente. L'URL HTTP dove la Relying Party fornisce il Request Object firmato all'Istanza del Wallet.
request_uri_method	OPZIONALE solo se `request_uri` è presente, altrimenti NON DEVE essere presente. Il metodo HTTP DEVE essere impostato con `get` o `post` (RPR-07, RPR-08, RPR-09). L'Istanza del Wallet dovrebbe utilizzare questo metodo per ottenere il Request Object firmato all'indirizzo fornito dal `request_uri`. Se non fornito o uguale a `get`, l'Istanza del Wallet DEVE utilizzare il metodo HTTP `get`. Altrimenti, l'Istanza del Wallet DOVREBBE fornire i suoi metadata all'interno del body della richiesta HTTP POST codificati in `application/x-www-form-urlencoded`.

Nota

Le specifiche IT Wallet raccomandano l'uso di `request_uri`, ovvero Request Object by reference.

Avvertimento

Per motivi di sicurezza e per prevenire attacchi di tipo endpoint mix-up, il valore contenuto nel parametro `request_uri` DEVE essere uno di quelli attestati da una terza parte fidata, come quelli forniti nei metadata `openid_credential_verifier` all'interno del parametro `request_uris`, ottenuti dalla Trust Chain relativa alla Relying Party (*WP_081* (pagina 798) and RPR-85).

Il valore corrispondente all'endpoint `request_uri` DOVREBBE essere casuale, secondo quanto prescritto da RFC 9101, The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR) Sezione 5.2.1.

Richiesta all'Endpoint URI Request

La Relying Party DOVREBBE abilitare il metodo POST nel suo Endpoint `request_uri` consentendo all'Istanza del Wallet di informare la Relying Party sulle sue capacità tecniche.

Questa funzionalità può essere utile quando, ad esempio, l'Istanza del Wallet supporta un insieme ristretto di funzionalità, algoritmi o un URL specifico per il suo `authorization_endpoint`, e qualsiasi altra informazione che ritiene necessario fornire alla Relying Party per l'interoperabilità.

Avvertimento

L'Istanza del Wallet, quando fornisce le sue capacità tecniche alla Relying Party, NON DEVE includere alcuna informazione dell'Utente o altre informazioni esplicite (RPR-86) riguardanti l'hardware utilizzato o le preferenze di utilizzo del suo Utente (RPR-86).

Se sia la Relying Party che l'Istanza del Wallet supportano il `request_uri_method` con HTTP POST, le capacità (metadata) dell'Istanza del Wallet DEVONO essere fornite utilizzando una richiesta HTTP all'endpoint `request_uri` della Relying Party, con il metodo POST e il tipo di contenuto impostato su *application/x-www-form-urlencoded* (*WP_083* (pagina 798)). La richiesta e i suoi parametri sono definiti nella Sezione 5 (Authorization Request) di Open ID4VP. Di seguito sono riportati i dettagli normativi e i riferimenti sui parametri da utilizzare dall'Istanza del Wallet nella Richiesta URI Request (*WP_083a–WP_083c* (pagina 798)).

Tabella 12.17: Parametri dell'Endpoint URI Request

Parametro	Descrizione
wallet_metadata	OPZIONALE. JSON Object con parametri di metadata. Vedi OpenID4VP, Sezione 10.1 e la tabella seguente, "Parametri dei metadata del Wallet".
wallet_nonce	RACCOMANDATO. Stringa utilizzata dall'Istanza del Wallet per prevenire il replay delle risposte della Relying Party.

Tabella 12.18: Parametri dei metadata del Wallet

Parametro	Descrizione
vp_formats_supported	OBBLIGATORIO. Oggetto contenente un elenco di coppie nome/valore, in cui il nome è un identificatore di formato di Credenziale e il valore definisce i parametri specifici del formato supportati da un Wallet. Vedere OpenID4VP Appendice B. Le Istanze del Wallet DEVONO supportare gli identificatori di formato di Credenziale richiesti da OPENID4VC-HAIP (inclusi `dc+sd-jwt` e `mso_mdoc`).
client_id_prefixes_supported	RACCOMANDATO. Un array non vuoto dei prefissi dell'identificatore del Client supportati dall'Istanza del Wallet. I valori validi includono `openid_federation` e `x509_hash`; se omesso, il valore predefinito è pre-registrato.

continues on next page

Tabella 12.18 – continua dalla pagina precedente

Parametro	Descrizione
authorization_endpoint	URL dell'endpoint del server di autorizzazione, vedi OAUTH2. L'utilizzo di un link universale è preferibile per una sicurezza migliorata e supporto di fallback, *URL schemes* personalizzati possono anche essere utilizzati se necessario.
response_types_supported	OPZIONALE. Array JSON di valori "response_type" di OAuth 2.0. Se presente DEVE essere impostato su `vp_token` (RPR-82).
request_object_signing_alg_values_supported	OPZIONALE. Vedi OpenID Connect Discovery.

Nota

Nell'IT Wallet, le Relying Party legacy che utilizzano un URI `https` come `client_id` seguono implicitamente il prefisso dell'identificatore del client previsto da OpenID Federation (`openid_federation`). La loro fiducia è stabilita e validata tramite la risoluzione della catena di fiducia, che è considerata equivalente a quella dei client fidati staticamente (`pre-registered`), come definito in [**RFC 6749**], per garantire la compatibilità con le versioni precedenti.

Nota

Il parametro `wallet_nonce` è RACCOMANDATO per le Istanze del Wallet che vogliono prevenire il *replay* delle loro richieste HTTP alle Relying Party da parte di avversari. Quando presente, la Relying Party DEVE controllarlo (RPR-81).

Nota

Per l'`authorization_endpoint` l'uso di *univarsal link* è preferito rispetto a *URL scheme* personalizzati perché, quando configurati correttamente utilizzando Assetlinks JSON per Android e Apple App Site Association per iOS, forniscono una sicurezza migliorata riducendo il rischio di dirottamento degli URL. Inoltre, gli *univarsal link* offrono meccanismi di fallback, consentendo al flusso di continuare senza problemi in un browser anche se l'Istanza del Wallet non è installata, garantendo un'esperienza Utente più fluida. Gli *URL scheme* `openid4vp://` e `haip-vp://` definiti in OPENID4VC-HAIP e OPENID4VP sono supportati per garantire l'interoperabilità.

Risposta dell'Endpoint URI Request

La Relying Party emette il Request Object firmato utilizzando il tipo di contenuto impostato su `application/oauth-authz-req+jwt`. Per il contenuto dell'oggetto Request, vedere la Sezione *Request Object* (pagina 525).

Errori dell'Endpoint URI Request

Quando la Relying Party incontra errori durante l'emissione del Request Object dall'endpoint `request_uri`, DEVE restituire una *Error Response* con `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

La seguente tabella elenca gli *HTTP Status Code* e i relativi *Error codes* che DEVONO essere supportati per la *Error Response*:

Codice di Stato	Codice di Errore	Descrizione
`400 Bad Request`	`invalid_request`	L'Oggetto di Richiesta non può essere recuperato a causa di una richiesta non valida o malformata all'endpoint `request_uri`. (**RFC 6749#section-4.1.2.1**)
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint URI Request ha incontrato un problema interno. (**RFC 6749#section-4.1.2.1**).
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint URI Request è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico). (**RFC 6749#section-4.1.2.1**).

Di seguito è riportato un esempio di *Error Response* dall'endpoint `request_uri`:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
  "error": "server_error",
  "error_description": "The Request Object cannot be retrieved due to an internal server error.
"
}
```

Dopo aver ricevuto una *Error Response*, l'Istanza del Wallet DOVREBBE informare l'Utente della condizione di errore in modo appropriato (*WP_089* (pagina 798)). L'Istanza del

Wallet DOVREBBE registrare l'errore e PUÒ tentare di recuperare da determinati errori se fattibile (*WP_089a* (pagina 798)). Ad esempio, se l'errore è `server_error`, l'Istanza del Wallet DOVREBBE chiedere all'Utente di reinserire o scansionare un nuovo codice QR, se possibile (*WP_089b* (pagina 798)).

Request Object

I parametri dell'header JWT sono descritti di seguito:

Nome	Descrizione
alg	OBBLIGATORIO. Algoritmo utilizzato per firmare il JWT, secondo [**RFC 7516#section-4.1.1**]. DEVE essere uno degli algoritmi supportati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` o su un identificatore di algoritmo simmetrico (MAC) (RPR-88).
typ	OBBLIGATORIO. Media Type del JWT, come definito in [**RFC 7519**] e [**RFC 9101**]. DOVREBBE essere impostato sul valore `oauth-authz-req+jwt` (RPR-89).
kid	OBBLIGATORIO. ID della chiave della chiave pubblica necessaria per verificare la firma JWT, come definito in [**RFC 7517**].
trust_chain	OPZIONALE. È una sequenza di Entity Statement che compongono la Trust Chain relativa alla Relying Party, come definito in OID-FED Sezione 4.3 *Trust Chain Header Parameter*.
x5c	OBBLIGATORIO quando `client_id` utilizza uno schema con prefisso `x509_hash`. OPZIONALE quando `client_id` utilizza lo schema `openid_federation`. Contiene il certificato X.509 foglia della Relying Party (e opzionalmente i certificati intermedi), utilizzato per verificare la firma JWT con la chiave pubblica nel certificato della Relying Party come definito in **RFC 7515**. Il certificato della Relying Party in `x5c` DEVE attestare informazioni di identità della Relying Party sufficienti per vincolare gli endpoint di rete referenziati dal flusso di presentazione. In particolare, gli endpoint utilizzati nella Authorization Request e Authorization Response (ad esempio, `response_uri`, `redirect_uri`) DEVONO corrispondere alle informazioni di identità contenute nel certificato della Relying Party (ad esempio, un SAN di tipo URI per il matching completo dell'URI o un DNS Name SAN per il matching del nome host).

Nota

L'intestazione `x5c` NON DEVE includere il certificato radice, come richiesto da OPENID4VC-HAIP. La catena di certificati `x5c` DEVE validare a un certificato radice preconfigurato;

vedere la Sezione *X.509 PKI* (pagina 147) per informazioni di base sulla validazione della catena di certificati X.509.

I parametri del payload JWT sono descritti qui:

Nome	Descrizione
client_id	OBBLIGATORIO. Identificatore univoco della Relying Party.
client_metadata	**OBBLIGATORIO. Un oggetto JSON contenente i valori dei metadata della Relying Party come definito nella sezione 5.1 di OpenID4VP, che DOVREBBE includere i seguenti parametri:** • **vp_formats_supported**. Utilizzato dall'Istanza del Wallet per determinare i formati di Verifiable Presentation supportati. • **encrypted_response_enc_values_supported**. Array JSON che elenca gli algoritmi JWE `enc` supportati per le Authorization Response cifrate in `direct_post.jwt`. • **jwks**. JSON Web Key Set utilizzato dall'Istanza del Wallet per cifrare la Authorization Response o per l'accordo delle chiavi. Le chiavi contenute in questo set sono specifiche della richiesta e identificate dal loro valore `kid`. • **client_name** e **logo_uri**. OPZIONALE. Utilizzati per la visualizzazione del consenso dell'utente e per mostrare l'identità della Relying Party nell'interfaccia dell'Istanza del Wallet.
response_mode	OBBLIGATORIO. DEVE essere impostato su `direct_post.jwt` sia nel Flusso Same Device sia nel Flusso Cross Device (RPR-90).
dcql_query	OBBLIGATORIO. Oggetto che rappresenta una richiesta di presentazione di Credenziali, secondo il linguaggio di query DCQL definito nella Sezione 6 di OpenID4VP.

continues on next page

Tabella 12.21 – continua dalla pagina precedente

Nome	Descrizione
transaction_data	**OPZIONALE. Un array non vuoto di oggetti JSON, ognuno dei quali descrive una transazione che la Relying Party richiede all'Utente di autorizzare. Ogni oggetto di transazione include:** • **type**. Stringa che identifica il tipo di dati della transazione. • **credential_ids**. Array che fa riferimento a una o più Credenziali provenienti dalla `dcql_query` che possono autorizzare la transazione.
transaction_data_hashes_alg	OPZIONALE. Un array di stringhe, ciascuna delle quali rappresenta un identificatore di algoritmo di hash, corrispondente a un nome di algoritmo di hash elencato nel registro IANA. Uno di questi algoritmi DEVE essere utilizzato per calcolare gli hash nel parametro di risposta `transaction_data_hashes`. Se omesso, l'algoritmo di hash predefinito è sha-256.
response_type	OBBLIGATORIO. DEVE essere impostato su `vp_token` (RPR-91).
wallet_nonce	OBBLIGATORIO se precedentemente fornito dall'Istanza del Wallet (RPR-81). Valore stringa utilizzato per mitigare gli attacchi di replay della Request URI Response, come definito nella Sezione 5.10 (Request URI Method) di OpenID4VP.
response_uri	OBBLIGATORIO. L'URI di Risposta a cui l'Istanza del Wallet DEVE inviare la Authorization Response utilizzando una HTTP Request con il metodo POST (RPR-92).
nonce	OBBLIGATORIO. Numero unico e casuale con sufficiente entropia, la cui lunghezza DEVE essere di almeno 32 cifre (RPR-93)..
state	RACCOMANDATO. Identificatore univoco della Authorization Request, il suo valore DOVREBBE essere opaco per l'Istanza del Wallet.
iss	OBBLIGATORIO. L'entità che ha emesso il JWT. Sarà popolato con il `client_id` della Relying Party.
iat	OBBLIGATORIO. Timestamp Unix, che rappresenta l'ora in cui il JWT è stato emesso.
exp	OBBLIGATORIO. Timestamp Unix, che rappresenta l'ora di scadenza in cui o dopo la quale il JWT NON DEVE più essere valido (RPR-94).

Avvertimento

Per motivi di sicurezza e per prevenire attacchi di tipo endpoint mix-up, il valore contenuto nel parametro `response_uri` DEVE essere uno di quelli attestati da una terza parte fidata, come

quelli forniti nei metadata `openid_credential_verifier` all'interno del parametro `response_uris`, ottenuti dalla Trust Chain relativa alla Relying Party (*WP_091a* (pagina 798) and RPR-95).

Nota

Il parametro `transaction_data` è destinato ai casi d'uso in cui l'Istanza del Wallet DEVE autorizzare una transazione specifica, come l'avvio di un pagamento o una firma digitale. In questi scenari ad alta sensibilità, l'obiettivo è vincolare i dettagli della transazione alla Authorization Response, in modo da preservarne l'integrità e consentire di dimostrare successivamente l'approvazione dell'Utente (non ripudio).

Il meccanismo di binding dipende dal formato della Credenziale:

- **dc+sd-jwt**: il Wallet vincola i dati della transazione restituendo `transaction_data_hashes` (e, quando applicabile, `transaction_data_hashes_alg`) all'interno del Key Binding JWT (KB-JWT). Per ulteriori dettagli, vedere OpenID4VP Appendice B.3.3.
- **mso_mdoc**: i dati della transazione sono vincolati tramite l'autenticazione del dispositivo mdoc. Per questo formato, il Wallet DEVE verificare che il `type` dei dati della transazione richiesti sia supportato dal tipo di documento e autorizzato dall'emittente (KeyAuthorizations). In caso contrario, il Wallet DEVE rifiutare la richiesta a causa di un tipo di dati della transazione non supportato. Per ulteriori dettagli, vedere Open ID4VP Appendice B.2.1.

Nota

Il parametro `state` in una richiesta OAuth è opzionale, ma è altamente RACCOMANDATO. Viene utilizzato principalmente per prevenire attacchi Cross-Site Request Forgery (CSRF) includendo un valore unico e imprevedibile che la Relying Party può verificare al momento della ricezione della risposta. Inoltre, aiuta a mantenere lo stato tra Request e Response, come le informazioni di sessione o altri dati di cui la Relying Party ha bisogno dopo il processo di autorizzazione.

Nota

L'utilizzo del parametro `client_metadata` è condizionale. Se `client_id` utilizza il prefisso `x509_hash`, tutti i metadata della Relying Party, oltre alla sua chiave pubblica utilizzata per firmare il Request Object, DEVONO essere forniti in `client_metadata`. Tuttavia, se il parametro è presente e `client_id` utilizza il prefisso `openid_federation`, l'Istanza del Wallet

DEVE ottenere i metadata della Relying Party attraverso la Trust Chain OpenID Federation (RPR-96) e NON DEVE utilizzare `client_metadata` per sovrascrivere o sostituire i metadata risolti. L'unica eccezione è `client_metadata.jwks` (e i relativi parametri sulle capacità di cifratura della risposta, come `encrypted_response_enc_values_supported`), che PUÒ essere utilizzato esclusivamente per trasportare chiavi pubbliche specifiche della richiesta (effimere) per cifrare la Authorization Response in `direct_post.jwt` (vedere OpenID4VP Sezione 8.3).

Authorization Response

Dopo aver ottenuto l'autorizzazione e il consenso dell'Utente per la presentazione degli Attestati Elettronici, l'Istanza del Wallet invia la Authorization Response all'endpoint `response_uri` della Relying Party utilizzando una richiesta HTTP con il metodo POST (*WP_091* (pagina 798)). Il contenuto della risposta DEVE essere cifrato secondo il profilo ad alta affidabilità definito in *OPENID4VC-HAIP'*, utilizzando la modalità di risposta 'direct_post.jwt* come previsto dalla Sezione 8.3 di *OpenID4VP*. Tale cifratura richiede l'uso dell'accordo di chiave ECDH-ES sulla curva P-256 e della cifratura del contenuto AES-GCM (*A128GCM* oppure *A256GCM*, preferendo *A256GCM* quando entrambi sono disponibili), utilizzando la chiave pubblica specifica della richiesta della Relying Party selezionata da *client_metadata.jwks* e identificata dal relativo *kid* (*WP_092* (pagina 798)). La chiave pubblica usata per criptare l'Authorization Response viene recuperata dal Wallet dal JWKs presente nel `client_metadata`. In accordo alla sezione 14. 5 di OpenID4VP è RACCOMANDATO l'utilizzo di chiavi effimere.

Nota

Perché la risposta è cifrata?

La risposta inviata dall'Istanza del Wallet alla Relying Party è cifrata per impedire a un avversario di sfruttare possibili vulnerabilità per accedere alle informazioni trasmesse in chiaro all'interno della rete della Relying Party. Per esempio, ciò è possibile se l'ambiente di rete della Relying Party impiega un proxy per le operazioni di TLS Termination, il quale agisce come intermediario tra il client e il backend web server della Relying Party e gestisce tutte le operazioni relative a TLS. In questo caso specifico, il proxy decifra il contenuto della trasmissione, in seguito lo inoltra al backend web server della Relying Party. Questa operazione può avvenire in chiaro oppure negoziando una ulteriore sessione TLS con il web server della Relying Party (sempre raccomandato). Nel primo caso, trasmissione dei dati TLS in chiaro, qualsiasi avversario all'interno del segmento di rete fra proxy e web server backend che intercettasse i dati trasmessi, potrebbe ottenere informazioni sensibili; se però la risposta è cifrata, la fattispece descritta viene mitigata anche mandando i dati in chiaro.

Nella Authorization Response vengono utilizzati i seguenti parametri (*WP_093* (pagina 798)):

Nome	Descrizione
vp_token	Questo oggetto DEVE contenere gli Attestati Elettronici presentati, indicizzati in base ai valori `id` delle Credenziali presenti nella `dcql_query` della Authorization Request. Il `vp_token` DEVE essere un oggetto JSON in cui ogni chiave corrisponde all'identificativo di una Credenziale richiesta e ogni valore è una singola presentazione oppure un array contenente una o più presentazioni relative a quella Credenziale. La codifica di ciascuna presentazione dipende dal formato della Credenziale, ad esempio: • **dc+sd-jwt**: una stringa SD-JWT VC (incluso il Key Binding JWT aggiunto in coda) (*WP_093a* (pagina 798)). • **mso_mdoc**: un `DeviceResponse` CBOR codificato in base64url corrispondente alla presentazione mdoc richiesta (vedere OpenID4VP Appendice B.2). Quando vengono restituite più presentazioni mdoc, ciascuna DEVE essere trasportata in un `DeviceResponse` separato, allineato al corrispondente elemento della query DCQL; in tal caso, il valore di `vp_token` per quell'identificativo di Credenziale DEVE essere un array di valori `DeviceResponse`.
state	Identificatore univoco fornito dalla Relying Party all'interno della Authorization Request.

Nota

Sebbene OpenID4VP prenda in considerazione la bozza -10 della specifica SD-JWT VC, la specifica IT Wallet considera la bozza -11 (SD-JWT-VC) in linea con la versione identificata in OpenID4VCI.

SD-JWT definisce come un *Holder* può presentare una Attestato Elettronico a una Relying Party, dimostrando il legittimo possesso dell'Attestato Elettronico. Per fare ciò, l'*Holder* DEVE includere il `KB-JWT` nell'SD-JWT aggiungendo il `KB-JWT` alla termine della stringa contenente l'SD-JWT (*WP_093b* (pagina 798)), come rappresentato nell'esempio seguente

```
<Issuer-Signed-JWT>~<Disclosure 1>~<Disclosure 2>~...~<Disclosure N>~<KB-JWT>
```

Per convalidare la firma sul Key Binding JWT, la Relying Party DEVE utilizzare il materiale crittografico incluso nell'*Issuer-Signed-JWT*. La convalida della firma del Key Binding JWT (KB-JWT) DEVE utilizzare la chiave pubblica inclusa nell'SD-JWT contenuta nel parametro `cnf` contenuto nell'*Issuer-Signed-JWT*.

Quando viene presentato un SD-JWT, il suo KB-JWT DEVE contenere i seguenti parametri nell'intestazione JWT (*WP_093c* (pagina 798)):

Claim	Descrizione
typ	OBBLIGATORIO. DEVE essere `kb+jwt`, che caratterizza esplicitamente il Key Binding JWT come raccomandato nella Sezione 3.11 di **RFC 8725**.
alg	OBBLIGATORIO. Algoritmo di Firma utilizzando uno di quelli specificati nella Sezione *Algoritmi Crittografici* (pagina 647).

Quando viene presentato un SD-JWT, la firma KB-JWT DEVE essere verificata dalla stessa chiave pubblica inclusa nell'SD-JWT all'interno del parametro `cnf`. Il KB-JWT DEVE contenere i seguenti parametri nel payload JWT:

Claim	Descrizione
iat	OBBLIGATORIO. Il valore di questa *claim* DEVE essere l'ora in cui è stato emesso il Key Binding JWT, utilizzando la sintassi definita in **RFC 7519**.
aud	OBBLIGATORIO. Il ricevitore previsto del Key Binding JWT. Il valore di questo parametro DEVE corrispondere all'identificatore di entità (`client_id`) univoco della Relying Party.
nonce	OBBLIGATORIO. Garantisce l'unicità della firma. Il valore di questa *claim* DEVE essere una stringa e deve corrispondere a quello fornito nel Request Object.
sd_hash	OBBLIGATORIO. Il digest codificato in base64url del JWT firmato dal Fornitore di Attestati Elettronici (SD-JWT) e le *selective disclosures* selezionate dall'Utente.
transaction_data_hashes	CONDIZIONALE. OBBLIGATORIO quando la richiesta include `transaction_data`. Array non vuoto di hash codificati in base64url. Ogni hash è calcolato sul valore esatto della stringa corrispondente all'elemento `transaction_data`.
transaction_data_hashes_alg	CONDIZIONALE. OBBLIGATORIO solo se la richiesta includeva `transaction_data_hashes_alg`. Stringa che indica l'algoritmo di hash effettivamente utilizzato per calcolare `transaction_data_hashes`; se tale parametro non è stato fornito, la funzione di hash DEVE essere `sha-256`.

Errori della Authorization Response

Ci sono casi in cui l'Istanza del Wallet non può convalidare il Request Object o il Request Object risulta non valido. Questo errore si verifica se il Request Object viene recuperato con successo dall'url fornito nel parametro `request_uri` ma non supera i controlli di convalida. Ciò potrebbe essere dovuto a firme errate, claim non conformim o altri errori di convalida, come la revoca della trust per la specifica Relying Party che lo ha emesso.

Se l'Istanza del Wallet incontra tali errori durante la valutazione della Authorization Request, DEVE notificare alla Relying Party inviando una *Error Response* nella *Authorization Response* (*WP_090* (pagina 798)). L'Istanza del Wallet invia la *Error Response* nella *Authorization Response* all'endpoint `response_uri` della Relying Party utilizzando una richiesta HTTP POST (*WP_090* (pagina 798)). Questa *Error Response* DEVE essere codificata nel corpo della richiesta utilizzando il formato definito dal tipo di contenuto `application/x-www-form-urlencoded`.

Di seguito è riportato un esempio non normativo di una *Error Response* nella *Authorization Response*.

```
POST /response_uri HTTP/1.1
HOST: relying-party.example.org
Content-Type: application/x-www-form-urlencoded

state=3be39b69-6ac1-41aa-921b-3e6c07ddcb03&
error=invalid_request&
error_description=...
```

Avvertimento

L'attuale specifica OpenID4VP delinea varie risposte di errore che un'Istanza del Wallet può restituire alla Relying Party in caso di *Authorization Request* errata. Per migliorare la privacy, le Istanze del Wallet NON DOVREBBERO notificare alla Relying Party le richieste errate qualora un uso improprio delle risposte di errore potrebbe portare a raccogliere informazioni lesive della privacy dell'Utente (ad esempio, se l'Utente decide di non voler presentare l'Attestato Elettronico richiesto).

Nella seguente tabella sono elencati gli *Error codes* e le descrizioni che sono supportati per la *Error Response* nella *Authorization Response*:

Codice di Errore	Descrizione
`invalid_request_uri`	Il *request_uri* nella Authorization Request restituisce un errore, contiene dati non validi o è altrimenti malformato. **RFC 9101**

continues on next page

Tabella 12.25 – continua dalla pagina precedente

Codice di Errore	Descrizione
`vp_formats_not_supported`	L'Istanza del Wallet non supporta nessuno dei formati vp richiesti dalla Relying Party. OpenID4VP
`invalid_request_uri_method`	Il valore del parametro `request_uri_method` non è né `get` né `post`. OpenID4VP
`invalid_request`	La richiesta è malformata o incoerente (ad esempio utilizza il Response Type `vp_token` ma non include il parametro `dcql_query`), il prefisso dell'identificatore del Client non è supportato oppure i requisiti del prefisso non sono rispettati (ad esempio, `client_id` con il prefisso `x509_hash` senza il `client_metadata` richiesto). OpenID4VP
`access_denied`	Il Wallet non aveva l'Attestato Elettronico richiesto, l'Utente non ha dato il consenso o il Wallet non è riuscito ad autenticare l'Utente. OpenID4VP
`invalid_client`	• I metadata della Relying Party sono stati risolti basandosi sull'Identificatore del Client (utilizzando il prefisso `openid_federation` o `x509_hash`), ma la Relying Party non può essere autorizzata a causa di errori nella verifica della trust oppure dal fatto che non è stata riconosciuta come entità valida della federazione. OID-FED e Open ID4VP
`invalid_transaction_data`	Uno o più oggetti nella struttura `transaction_data` non sono validi. Ad esempio, tali oggetti contengono tipi sconosciuti o non supportati, campi malformati (ad esempio è un oggetto di tipo noto ma contiene campi sconosciuti o campi di tipo errato per il tipo di transaction data) o mancanti, valori non validi (ad esempio il campo `credential_ids` non corrisponde) oppure riferimenti a Credenziali non disponibili. OpenID4VP

Risposta della Relying Party

Come definito nella Sezione 8.2. (*Response mode* "direct_post") della specifica OpenID4VP, se l'Endpoint di Risposta della Relaying Party ha elaborato con successo la *Authorization Response* o la *Error Response* fornita dall'Istanza del Wallet, DEVE rispondere con un codice di stato HTTP 200 con `Content-Type` di `application/json` e un JSON Object nel corpo della risposta.

Nel **Flusso Same Device**, la Relying Party DEVE aggiungere il parametro `redirect_uri` all'JSON Object nel corpo della risposta. Dopo aver ricevuto il `redirect_uri`, l'Istanza del Wallet DEVE eseguire un reindirizzamento all'URL specificato dal `redirect_uri`. Questo reindirizzamento consente alla Relying Party di riprendere senza problemi l'interazione con l'Utente sul dispositivo che ha avviato il flusso, dopo che l'Istanza del Wallet ha trasmesso la Authorization Response al `response_uri` designato.

La Relying Party DEVE includere un codice di risposta all'interno del `redirect_uri`. Il codice di risposta è un numero fresco, crittograficamente casuale utilizzato per garantire che solo il

ricevitore del reindirizzamento possa recuperare ed elaborare la Authorization Response. Il numero potrebbe essere aggiunto come componente del path, come parametro o come frammento all'URL. È RACCOMANDATO utilizzare un valore casuale crittografico di 128 bit o più al momento della scrittura di questa specifica. Anche se un avversario riesce a rubare il valore casuale utilizzato nella richiesta allo Status Endpoint, il suo user-agent verrebbe rifiutato a causa del cookie mancante nella richiesta.

Avvertimento

Per motivi di sicurezza e per prevenire attacchi di tipo endpoint mix-up, il valore contenuto nel parametro `redirect_uri` DEVE essere uno di quelli attestati da una terza parte fidata, come quelli forniti nei metadata `openid_credential_verifier` all'interno del parametro `redirect_uris`, ottenuti dalla Trust Chain relativa alla Relying Party (*WP_094a* (pagina 798)).

Errori della Risposta della Relying Party

Se qualsiasi controllo di convalida, eseguito dalla Relying Party sulla Authorization Response dall'Istanza del Wallet, fallisce; l'endpoint URI di Risposta DEVE restituire una *Error Response*. La struttura di questa *Error Response* dovrebbe essere determinata dalla natura specifica dell'errore incontrato. La risposta DEVE utilizzare `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

La seguente tabella elenca gli Status Code HTTP e i relativi *Error codes* che DEVONO essere supportati per la *Error Response*:

Codice di Stato	Codice di Errore	Descrizione
`400 Bad Request`	`invalid_request`	La risposta non può essere elaborata perché mancano parametri obbligatori, contiene parametri non validi o è non conforme agli standard.
`400 Bad Request`	`invalid_request`	Gli Attestati Elettronici presentati sono non conformi agli standard, non valide o revocate.
`400 Bad Request`	`invalid_request`	La presentazione degli Attestati Elettronici, contenuta nell'oggetto `vp_token`, non è conforme agli standard, non ha i parametri richiesti o è formattata in modo errato.
`400 Bad Request`	`invalid_request`	L'"sd-jwt" restituito non è conforme agli standard, mancano parametri obbligatori o è formattato in modo errato.

continues on next page

Tabella 12.26 – continua dalla pagina precedente

Codice di Stato	Codice di Errore	Descrizione
403 Forbidden	invalid_request	La firma del KB-JWT non è valida o non corrisponde alla chiave pubblica associata (JWK) referenziata nell'SD-JWT firmato dall'Emittente.
403 Forbidden	invalid_request	Il valore del nonce fornito è errato o altrimenti non conforme agli standard.
403 Forbidden	invalid_request	La trust non può essere stabilita con il Fornitore di Attestati Elettronici.
500 Internal Server Error	server_error	La richiesta non può essere soddisfatta perché il Response Endpoint ha incontrato un problema interno.
503 Service Unavailable	temporarily_unavailable	La richiesta non può essere soddisfatta perché il Response Endpoint è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

Di seguito ci sono due esempi di risposte HTTP che utilizzano `application/json` che includono sia i membri `error` che `error_description`:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "Trust cannot be established with the issuer: https://issuer.example.com
"
}
```

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The vp_token is malformed, missing required parameters or incorrectly f
ormatted"
}
```

Status Endpoint

Questa specifica introduce lo Status Endpoint della Relying Party per le implementazioni che scelgono di utilizzarlo. Questo Endpoint è una funzionalità di sicurezza interna dell'implementazione e non è richiesto per l'interoperabilità.

Sia he il flusso sia Same Device o Cross Device, l'user-agent deve controllare lo stato della sessione all'endpoint reso disponibile dalla Relying Party (endpoint di stato). Questo controllo PUÒ essere implementato sotto forma di codice JavaScript, all'interno della pagina che mostra

il Codice QR o il tramite pulsante href che punta all'URL di richiesta. Il codice JavaScript fa sì che l'user-agent controlli lo Status Endpoint utilizzando una strategia di polling (in secondi) o una strategia push (ad esempio, WebSocket).

Poiché la pagina HTML e lo Status Endpoint sono implementati dalla Relying Party, i dettagli di implementazione di questa soluzione sono responsabilità della Relying Party, in quanto sono relativi alle API interne della Relying Party. Tuttavia, il testo seguente descrive un'implementazione di esempio.

La Relying Party lega la richiesta dello user-agent, con un cookie di sessione contrassegnato come `Secure` e `HttpOnly`, con la richiesta emessa. L'URL della richiesta DOVREBBE includere un parametro con un valore casuale. La risposta HTTP restituita da questo endpoint di stato PUÒ contenere gli Status Code HTTP elencati di seguito:

- **201 Created**; quando il Request Object firmato è stato emesso dalla Relying Party che attende di essere scaricato dall'Istanza del Wallet all'endpoint `request_uri`.
- **202 Accepted**; quando il Request Object firmato è stato ottenuto dall'Istanza del Wallet.
- **200 OK**; quando l'Istanza del Wallet ha fornito la presentazione all'endpoint `response_uri` della Relying Party e l'autenticazione dell'Utente ha avuto successo. La Relying Party aggiorna il cookie di sessione consentendo allo user-agent di accedere alla risorsa protetta. Viene fornito un `redirect_uri` che trasporta lo user-agent alla pagina in cui l'Utente deve navigare.

Errori dello Status Endpoint

Se invece qualsiasi controllo di convalida eseguito dalla Relying Party fallisce, la pagina del Codice QR DOVREBBE essere aggiornata con un messaggio di errore. Inoltre, lo Status Endpoint DEVE restituire una *Error Response*, la cui struttura dipende dalla natura dell'errore. La risposta DEVE utilizzare `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

La seguente tabella elenca gli Status Code HTTP e i relativi *Error codes* che DEVONO essere supportati per la *Error Response*:

Codice di Stato	Codice di Errore	Descrizione
`401 Unauthorized`	`authentication_failed`	L'Istanza del Wallet o il suo Utente hanno rifiutato la richiesta, la richiesta è scaduta o altri errori hanno impedito l'autenticazione.
`403 Forbidden`	`invalid_session`	L'id di sessione fornito nella richiesta non è valido.

URI di Reindirizzamento

Il valore `redirect_uri` DEVE essere utilizzato con un metodo HTTP GET dall'user-agent per reindirizzare l'Utente a un endpoint specifico della Relying Party al fine di completare il processo.

Errori dell'URI di Reindirizzamento

Quando l'user-agent viene reindirizzato all'URI di Reindirizzamento fornito dalla Relying Party, possono verificarsi diversi errori che impediscono il completamento con successo del processo. Questi errori sono critici in quanto influenzano direttamente l'esperienza dell'Utente ostacolando il flusso fluido di informazioni tra l'Istanza del Wallet e la Relying Party. La gestione di questi errori richiede una comunicazione chiara all'Utente all'interno della pagina web di navigazione restituita. La Relying Party DEVE implementare i meccanismi di gestione degli errori e di convalida per gli URI di Reindirizzamento definiti in questa specifica. Di seguito sono riportati potenziali errori relativi all'URI di Reindirizzamento, la *Error Response* DEVE utilizzare `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

La seguente tabella elenca gli Status Code HTTP e i relativi *Error codes* che DEVONO essere supportati per la *Error Response*:

Codice di Stato	Codice di Errore	Descrizione
`403 Forbidden`	`invalid_request`	L'URI di Reindirizzamento fornito dalla Relying Party non corrisponde a nessuno degli URI collegati alla sessione dell'Utente. (**RFC 6749#section-4.1.2.1**)
`403 Forbidden`	`invalid_request`	La sessione dell'Utente non è valida o è scaduta.
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta a causa di un errore interno del server. (**RFC 6749#section-4.1.2.1**).

continues on next page

Tabella 12.28 – continua dalla pagina precedente

Codice di Stato	Codice di Errore	Descrizione
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché il servizio è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico). (**RFC 6749#section-4.1.2.1**).

12.2.2 Flusso di Prossimità

Questa sezione descrive come un'Istanza di Relying Party richiede la presentazione di un Attestato Elettronico *mdoc-CBOR* a un'Istanza del Wallet come dettagliato nella *Specifica ISO 18013-5*.

La fase di presentazione di alto livello è strutturata in tre ampie sotto-fasi come illustrato nella figura seguente:

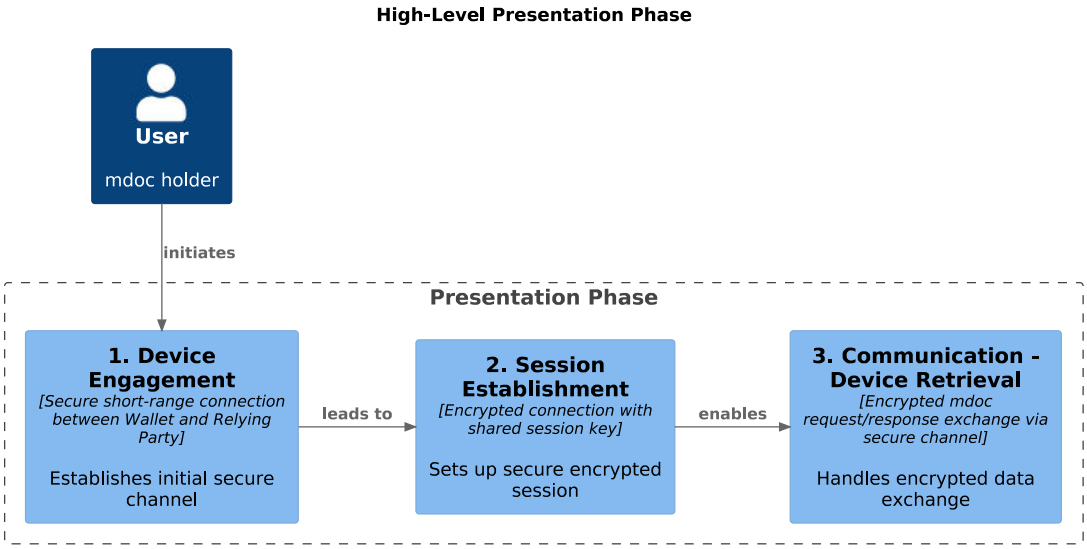


Fig. 12.14: Flusso di Presentazione di Alto Livello in prossimità.

Le sotto-fasi sono descritte di seguito:

1. **Device Engagement**: Questa sottofase inizia quando l'Utente viene invitato a divulgare attributi specifici dagli mdoc. L'obiettivo di questa sottofase è stabilire un canale di comunicazione sicuro tra l'Istanza del Wallet e l'Istanza di Relying Party, in modo che le richieste e le risposte mdoc possano essere scambiate durante la sottofase di comunicazione. I messaggi scambiati in questa sottofase sono trasmessi attraverso tecnologie a corto raggio per limitare la possibilità di monitorarli o intercettarli da parte di terzi. I dati di *Device Engagement* possono essere trasferiti utilizzando QR code o NFC.

2. **Session Establishment**: Durante la fase di *Session Establishment*, l'Istanza di Relying Party configura una connessione sicura. Tutti i dati trasmessi su questa connessione sono cifrati utilizzando una chiave di sessione, che è nota sia all'Istanza del Wallet che all'Istanza di Relying Party. La sessione stabilita PUÒ essere successivamente terminata in base alle condizioni dettagliate in [ISO18013-5 #12. 2.4].

1. **Comunicazione - Device Retrieval**: L'Istanza di Relying Party cifra la Richiesta mdoc con la chiave di sessione appropriata e la invia all'Istanza del Wallet insieme alla sua chiave pubblica nel messaggio di *Session Establishment*. L'Istanza del Wallet utilizza i dati del messaggio di *Session Establishment* per derivare la chiave di sessione e quindi per decifrare la Richiesta mdoc. Durante la sottofase di comunicazione, l'Istanza di Relying Party ha l'opzione di richiedere informazioni dall'Istanza del Wallet utilizzando richieste e risposte mdoc. La modalità principale di comunicazione è il canale sicuro stabilito durante la configurazione della sessione. L'Istanza del Wallet cifra la Risposta mdoc utilizzando la chiave di sessione e la trasmette all'Istanza di Relying Party mobile tramite un messaggio di *Session Data*.

Le Istanze di Relying Party e Wallet registrate nell'ecosistema IT-Wallet DEVONO supportare almeno:

- *Flusso di Recupero del Dispositivo Supervisionato* dove un supervisore umano supervisiona il processo di verifica in persona, in contrasto con il *flusso non supervisionato* dove la verifica potrebbe avvenire attraverso sistemi automatizzati senza supervisione umana (*WP_095* (pagina 798)).
- *Autenticazione dell'Istanza di Relying Party* seguendo i meccanismi definiti nella ISO18013-5 per il *reader authentication* (*WP_098* (pagina 798)).
- *Tipo di Documento* domestico e *Namespace* definiti in questa specifica tecnica in aggiunta a quelli già definiti nell'ISO18013-5 per l'mDL (vedi *Formato Attestato Elettronico mdoc-CBOR* (pagina 389) per maggiori dettagli) (*WP_099* (pagina 798)).

La tabella seguente mostra le tecnologie di *Device Engagement* supportate (*WP_097* (pagina 798)), specificando quali sono obbligatorie.

Tecnologia di trasmissione	ISO 18013-5 Istanza del Wallet	ISO 18013-5 Istanza di Relying Party	IT Wallet Istanza del Wallet	IT Wallet Istanza di Relying Party
QR code	C[a]	M	DEVE	C – DEVE se il dispositivo è dotato di una fotocamera o lettore QR code e BLE.

continues on next page

Tabella 12.29 – continua dalla pagina precedente

Tecnologia di trasmissione	ISO 18013-5 Istanza del Wallet	ISO 18013-5 Istanza di Relying Party	IT Wallet Istanza del Wallet	IT Wallet Istanza di Relying Party
NFC	C[a]	M	RACCOMANDATO	C – DEVE se il dispositivo è dotato di un lettore NFC.

Legenda: C = Condizionale | M = Obbligatorio | [a] Il supporto per almeno uno di questi metodi è obbligatorio (*WP_097a* (pagina 798))

La tabella seguente mostra le tecnologie di *Device Retrieval* supportate, specificando quali sono obbligatorie.

Tecnologia di trasmissione	ISO 18013-5 Istanza del Wallet	ISO 18013-5 Istanza di Relying Party	IT Wallet Istanza del Wallet	IT Wallet Istanza di Relying Party
BLE	C[a]	M	DEVE	C – DEVE se il dispositivo è dotato di una fotocamera o lettore QR code e BLE.
NFC	C[a]	M	RACCOMANDATO	C – DEVE se il dispositivo è dotato di un lettore NFC.

Legenda: C = Condizionale | M = Obbligatorio | [a] Il supporto per almeno uno di questi metodi è obbligatorio (*WP_096b* (pagina 798))

Nota

Dalla seconda edizione, versione 3, ISO18013-5 non definisce o supporta *Server Retrieval* come opzione di trasporto. Sono specificati solo metodi di recupero di prossimità (NFC, BLE e opzionalmente Wi-Fi Aware) (*WP_096* (pagina 798)). Pertanto, *Server Retrieval* non è considerato in questo flusso (*WP_096a* (pagina 798) and PPR-023).

La figura seguente illustra il flusso di basso livello conforme a ISO 18013-5 per il flusso di prossimità.

Passo 1: L'Utente apre l'Istanza del Wallet avviando il processo.

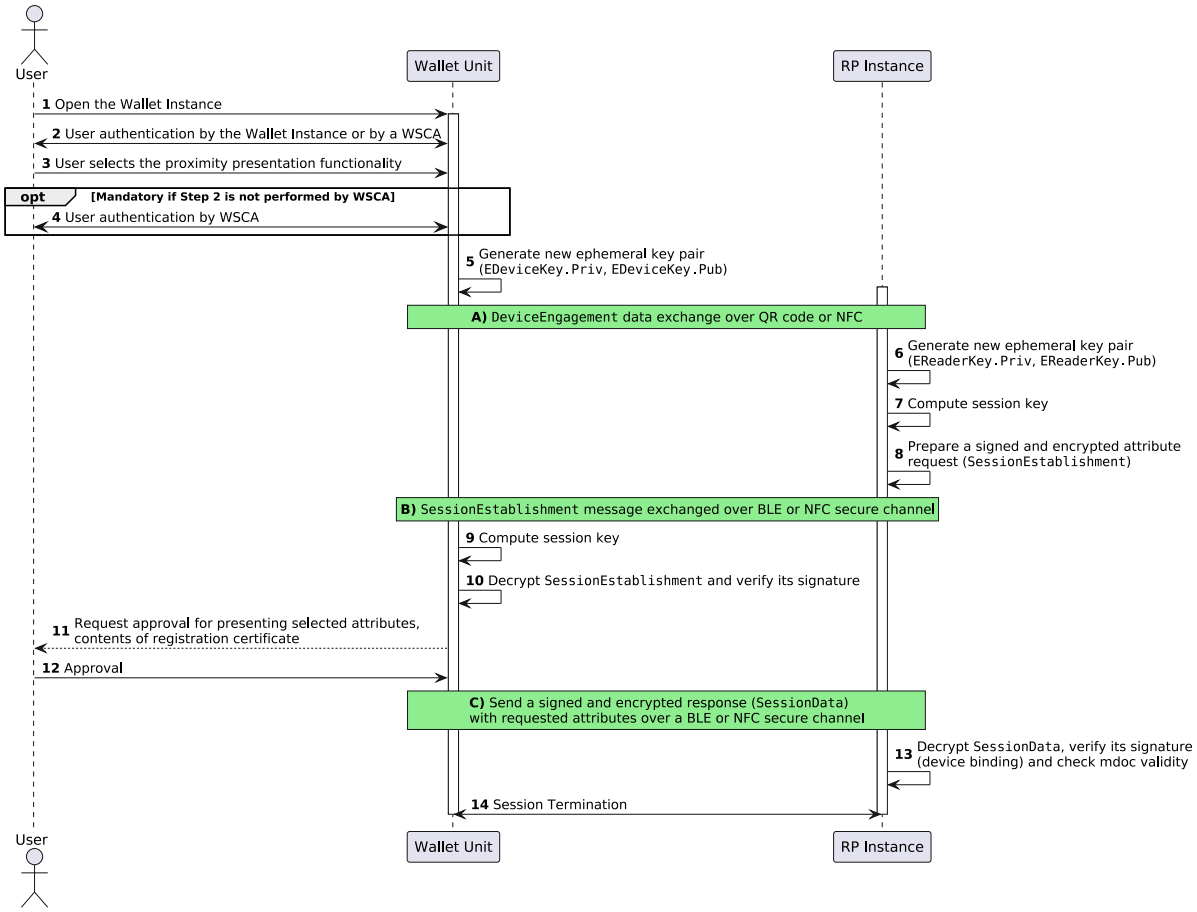


Fig. 12.15: Flusso di Presentazione di Basso Livello in prossimità.

Passo 2: L'Utente si autentica all'Istanza del Wallet. Questo può essere fatto dall'Istanza del Wallet o da un *Wallet Secure Cryptographic Application* (WSCA). È un prerequisito per accedere ai dati sensibili e presentare attributi (*WP_100* (pagina 798)).

Passo 3: L'Utente seleziona la funzionalità di presentazione di prossimità.

Passo 4: [Opzionale] Se l'autenticazione iniziale nel Passo 2 non è stata effettuata tramite WSCA, PUÒ essere richiesta un'autenticazione separata tramite WSCA (*WP_100* (pagina 798)).

Passo 5: L'Istanza del Wallet genera una nuova coppia di chiavi effimera per la comunicazione sicura (*WP_101* (pagina 798)). La chiave pubblica (`EDeviceKey.Pub`) sarà scambiata con l'Istanza di Relying Party per derivare una chiave di sessione condivisa, successivamente utilizzata per cifrare la sessione (processo di *Device Engagement*).

Box A

L'Istanza del Wallet e l'Istanza di Relying Party scambiano dati di *Device Engagement* tramite QR code o tramite NFC Connection Handover (*WP_097* (pagina 798)).

Fare riferimento a:

- Sez. 8.2.2.1 per `DeviceEngagement` tramite QR code
- Sez. 8.2.2.2 per `DeviceEngagement` tramite NFC

Passo 6: L'Istanza di Relying Party genera la sua coppia di chiavi effimera (`EReaderKey.Priv`, `EReaderKey.Pub`). La chiave privata (`EReaderKey.Priv`) DEVE essere mantenuta segreta, e la chiave pubblica (`EReaderKey.Pub`) DEVE essere utilizzata nel *Session Establishment*.

Passo 7: L'Istanza del Wallet e l'Istanza di Relying Party DEVONO derivare indipendentemente le chiavi di sessione utilizzando la loro chiave effimera privata e la chiave effimera pubblica dell'altra parte attraverso un Key Agreement Protocol opportuno. Questo garantisce lo scambio di messaggi cifrati nella sessione. In questo particolare passo, l'Istanza di Relying Party DEVE calcolare la sua chiave di sessione (PPR-002 and *WP_097* (pagina 798)).

Passo 8: L'Istanza di Relying Party DEVE preparare il `SessionEstablishment`. Questo messaggio DEVE essere firmato dall'Istanza di Relying Party (autenticazione dell'*mdoc reader* come specificato in [ISO18013-5 #12.5]) e cifrato utilizzando la chiave di sessione derivata nel passo precedente. Il messaggio `SessionEstablishment` DEVE includere la `EReaderKey.Pub` e una richiesta per specifici attributi (PPR-002).

Di seguito è riportato un esempio non normativo nella notazione diagnostica di un messaggio `SessionEstablishment` CBOR che contiene una Richiesta mdoc per un Attestato Elettronico mDL.

```
{
  "version": "1.0",  % Version
  "docRequests":
  [
    {
    "itemsRequest":
    24(<<  % Embedded CBOR data item
      {
        "docType": "org.iso.18013.5.1.mDL",
        "nameSpaces":
        {
          "org.iso.18013.5.1":
          {
            "family_name": true,
            "document_number": true,
            "driving_privileges": true,
            "issue_date": true,
            "expiry_date": true,
            "portrait": false
          }
        }
      }
    >>)
    }
  ],
  "readerAuthAll":
  [
    h'a10126', % COSE_Sign1 authentication header
    {
      33: h'308201253081cda00302010202012a300a06082a8648ce3d0403023020311e301c06035504030c1553
      6f6d65205265616465722041757468f72697479301e170d323331313234313023832325a170d3238313132
      323130323832325a301a311830160603550403c0f536f6d65205265616465722041b65793059301306072a86
      48ce3d020106082a8648ce3d03010703420004aa1092fb59e26ddd182cfdbc85f1aa8217a4f0fae6a6a5536b
      57c5ef7be2fb6d0dfd319839e6c24d087cd26499ec4f87c8c766200ba4c6218c74de50cd1243b'
    },
    null, % No additional reader authentication
    h'58a0d421a7e53b7db0412a196fea50ca6d4c8a530a47dd84d88588ab145374bd0ab2a724cf2ed2facf32c718
    4591c5969efd53f5aba63194105440bc1904e1b9'  % Reader authentication
    signature
  ]
}
```

Box B

L'Istanza di Relying Party DEVE trasmettere il messaggio `SessionEstablishment` cifrato e firmato all'Istanza del Wallet su una connessione NFC o BLE sicura stabilita sulla base delle

informazioni contenute nel Device Engagement. Fare riferimento a:

- Sez 8.2.2.3 per `SessionEstablishment` tramite BLE, e
- Sez 8.2.2.5 per `SessionEstablishment` tramite NFC

Passo 9: L'Istanza del Wallet DEVE calcolare la chiave di sessione, come descritto nel Passo 7.

Passo 10: Al ricevimento del `SessionEstablishment`, l'Istanza del Wallet DEVE decifrarlo utilizzando la chiave di sessione calcolata al Passo 9 e DEVE verificare la firma dell'Istanza di Relying Party (come specificato in [ISO18013-5 #12.5 *mdoc reader authentication*]) per garantire l'autenticità del messaggio (PPR-002 and *WP_105–106* (pagina 798)).

Passo 11: L'Istanza del Wallet DEVE decifrare la richiesta di attributi e DEVE chiedere all'Utente il consenso per il rilascio degli attributi richiesti (*WP_107* (pagina 798)). DEVE inoltre visualizzare il contenuto del Certificato di Registrazione della Relying Party per garantire l'autenticità, la trasparenza sui dati richiesti e sul suo scopo registrato (*WP_107a* (pagina 798)).

Passo 12: L'Utente esamina la richiesta e le informazioni di registrazione della Relying Party e quindi approva la presentazione degli attributi richiesti.

Box C

Dopo aver ricevuto l'approvazione dell'Utente, l'Istanza del Wallet DEVE recuperare gli Attestati Elettronici mdoc richiesti (PPR-006 and *WP_108* (pagina 798)). Deve quindi preparare un messaggio `SessionData` contenente questi Attestati Elettronici Digitali, e DEVE firmare i dati di autenticazione richiesti (come parte del processo di autenticazione mdoc, come specificato in [ISO18013-5 #12.4]) come specificato in (*WP_109–110* (pagina 798)). L'Istanza del Wallet DEVE cifrare il `SessionData` utilizzando la chiave di sessione stabilita prima di trasmetterlo all'Istanza di Relying Party tramite il canale sicuro (*WP_111* (pagina 798)). La firma garantisce il binding del dispositivo e l'integrità dei dati. La risposta mdoc DEVE essere codificata in CBOR, con la sua struttura delineata in [ISO18013-5 #10.3] (PPR-029, PPR-030, and *WP_112* (pagina 798)). Fare riferimento a (*WP_112a–112b* (pagina 798)):

- Sez 8.2.2.4 per `SessionData` tramite BLE, e
- Sez 8.2.2.5 per `SessionData` tramite NFC

Di seguito è riportato un esempio non normativo di `SessionData` nella notazione diagnostica che contiene la Risposta mdoc di un Attestato Elettronico mDL.

```
{
    "version": "1.0",
    "documents":
```

```
[
  {
    "docType": "org.iso.18013.5.1.mDL",
    "issuerSigned":
    {
      "nameSpaces":
      {
        "org.iso.18013.5.1":
        [
          24(<<
            {
                "digestID": 0,
                "random":
                h'790401ed5d0822d1aced942e4b0c41f754eee67b89c5ee3b8fd2c97491a96406',
                "elementIdentifier": "family_name",
                "elementValue": "Rossi"
            }
          >>),
          24(<<
            {
                "digestID": 3,
                "random":
                h'0c8f68d1ec3aa445ef68aa10b7a5875fa18ca222a821e23890a227cdc7d25e8f',
                "elementIdentifier": "issue_date",
                "elementValue": 1004(2025-03-27)
            }
          >>),
          24(<<
            {
                "digestID": 4,
                "random":
                h'0a9ed0d4937673152e52fb3fac0722baf4252e0d0c9869919e3339670203178e',
                "elementIdentifier": "expiry_date",
                "elementValue": 1004(2030-03-27)
            }
          >>),
          24(<<
            {
                "digestID": 8,
                "random":
                h'88e94c0365c611b523518d9a1b179ae52e242383576249f4965c40c6c97cf214',
                "elementIdentifier": "document_number",
                "elementValue": "XX1234567"
            }
          >>),
          24(<<
            {
```

```
"digestID": 9,
"random":
h'944758b43602b01ad68911b062349845492c04c6a78129bcf8cb5fb1396af2fc',
"elementIdentifier": "portrait",
"elementValue":
h'ffd8ffe000104a464946000101010090009000000ffdb004300130d0e110e0c
13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4c51685241
435f82
606871757b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434f
767676
76767676767676767676767676767676767676767676767676767676767676767676767676
767676
76767676767676ffc0001108001800640301220002110103110 1ffc4001b00000301000301
000000
0000000000000000050604010203 07ffc400321000010303030205020309000000000000001
020304
000511061221133114155161712241078 1a1163542527391b2c1f1ffc40015010101000000
000000
0000000000000000001ffc4001a1101010100030100000000000000000000000141112131
61ffda
000c03010002110311003f00a5bbde22da2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e770
9723a1
d0dae146ddfbb3c039ce07ad2bd47a7e32dbb8dd1d52d6ef4b284f64a480067dfb51f87ffb
95ff00
eb9ff14d215de66af089ce44b7dbde9cb6890a2838eddf18078f7add62d411ef4db9b10a65
d6b95a
147381ea0d495b933275fe6bba75c114104a8ba410413e983dff004f5af5d34b4b4cde632d
0bf1fd
1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e856001ebc03c7ce29dd9eef1e
f10fc4
47dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e54528750946e
c67048
50cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493
f602a0
75559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7
290875
db4be88ab384bbbbbfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30
ab420a
7a6d3240c718cf031fa9ef4c9ad550205aa02951df4a1d6c8421b015b769db8c9229837ea2
be8b1b
0d39d0eba9c51484efdb8c0efd8d258daf3c449699f2edbd4584e7af9c64e3f96b9beb28d4
ac4093
1e6478c8e76a24a825449501d867d2b1dcdebae99b9c752ae4ecd6dde4a179c1c1e460938f
9149ef
655e515c03919a289cb3dca278fb7bf177f4faa829dd8ce3f2ac9a7ecde490971fafd7dce1
5eed9b
```

```
            71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233dfec08238f6add62d391acc1c52
            56a79e
            706d52d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc71f39a7766b447b344
            e847dc
            ac9dcb5abba8d145061d43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca12a2c904a1
            5f71dd
            27d451453e09d1a21450961cbb3ea8a956433b781f1ce33dfed54f0e2b50a2b71d84ed6db1
            8028a2
            8175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a66b726e33de9ea6f1b52b181c7
            60724e
            47b514520a5a28a283ffd9'
        }
      >>),
      24(<<
        {
            "digestID": 10,
            "random":
            h'577e4822125f55fe923117aba01fdaefcc67d4aea80018fc22efa8d48e17982f',
            "elementIdentifier": "driving_privileges",
            "elementValue": [
              {
                "vehicle_category_code": "A",
                "issue_date": 1004("2020-09-17"),
                "expiry_date": 1004("2031-06-10")
                }
              ]
        }
      >>)
    ]
  },
  "issuerAuth":
  [
    << {1: -7} >>,
    {
      33: h'30820208308201afa00302010202142eb39c647c81836bcf79fa9cd0b201ec0bf52307300a0
      6082a8648ce3d0403023064310b3009060355040613025553311330110603550408 0c0a43616c6966
      6f726e6961311630140603550407 0c0d53616e204672616e636973636f31133011060355040a0c0a4
      d7920436f6d70616e7931133011060355040 30c0a6d79736974652e636f6d301e170d323530333237
      3135353532305a170d3235303430363135353532305a3064310b3009060355040613025553311330 1
      106035504080c0a43616c69666f726e6961311630140603550407 0c0d53616e204672616e63697363
      6f31133011060355040a0c0a4d7920436f6d70616e7931133011060355040 30c0a6d79736974652e6
      36f6d3059301306072a8648ce3d020106082a8648ce3d03010703420004f33da72d0dd0009b62221b
      0e839099b12dab5e01021124ebf9060422e648f3c3ec6614a86da1e91e552b2ae35e04d3058ae82b5
      c65a7f1f26800cb4499652a09a33f303d303b0603551d11043430328630 68747470733a2f2f637265
      64656e7469616c2d6973737565722e6f6964632d666564657261746 96f6e2e6f6e6c696e65300a060
      82a8648ce3d040302034700304402204d1f0819971652b79ebe4825547de3d5554d2f41410225e6b1
```

```
  3dab949cda125e022079ba71b823619e49719dce5daa565bf745d3d97e2b87c7f7d6a626f981e653
  ed'
},
<<
  24(<<
    {
      "version": "1.0",
      "digestAlgorithm": "SHA-256",
      "docType": "org.iso.18013.5.1.mDL",
      "valueDigests": {
        "org.iso.18013.5.1":
        {
            0: h'f46b65d5060ad060ab9be62ff22ea8633437619ebdc7fa81f2d151159e92bffe',
            1: h'e506545f6a6fd5d982670b4d62fc2b0688dc8f26754e7b0c574d63f5d72a85ac',
            2: h'cfcf96fa12d100eeed5f00183d3b6a0888baa47eae85b5b95037eca7bbc0d07e',
            3: h'8b0772252b0e06b611676b6b3402eb33bf866eb145e49f4d5f23215e6a047772',
            4: h'14135c96693e2ab08d956876ee491357d906a6dd125557196dfb9811ba54aa8d',
            5: h'86dcbd99233fbb84a9a2dce3a864a425e6e809300067a4475e3ea2a4d233dc74',
            6: h'2e9512d35ea225e69e7b2180ecc1678dcc3e77a16e36427e64b4f0e2861b4d3a',
            7: h'4efe55c36f6249d23c473a125afc5181aa3063393649478155497 1b72ff13700',
            8: h'cc44a4f9983c5b0b1efc0e82e2867c8d5bbdf89c34bff16a1953c923bb4e4b3e',
            9: h'775eb2af0aa55f2071d62662b35c99698ae3bc0e2c4af5724ff88476cddd152f',
           10: h'915d0ad53dd23dace34968c263d307c04701a9bb9dc9865af91dc409786fd833',
           11: h'47d89ff4fb513044e6f2394236755ac0abf3e4f4a46f40454a458a59f8b7a6fb'
        },
        "org.iso.18013.5.1.IT": {
           12: h'd665c50c4364c7cbf4ab9461b9bbb228f37ad278b9fb61283550951624d4d9ae',
           13: h'dc9d032a64866e33d06f48a882989b5747da3638f0d216a2275191ed3395fdec'
          }
      },
      "deviceKeyInfo": {
        "deviceKey": {
            1: 2,
           -1: 1,
           -2: h'f96b29873b61f05403e2963a7ecbc799c9aab28d8a6629e5848cfdef85442866',
           -3: h'a9fef033a900c63e3894d8deb805a2a1fb55ef0d2b88e3c0d3336408186485ef'
        }
      },
      "validityInfo": {
        "signed": 0("2025-03-27T00:00:00Z"),
        "validFrom": 0("2025-03-27T00:00:00Z"),
        "validUntil": 0("2026-03-27T00:00:00Z")
      },
      "status": {
        "status_list": {
          "idx": 1340,
          "uri": "https://statusprovider.example.org/statuslists/1"
```

```
                }
              }
            }
          >>)
        >>,
        h'd09f9acdf7a6be5e4aeb405bfb3b297b1b8003bcf52558a2f39fc6e5cffed40f18f49d2cc0e72a2a
        5645
        8d8aade591dee8d6540e639bca637f94bd9fa56f345c'
      ]
    },
    "deviceSigned":
    {
      "nameSpaces": 24(<< {} >>),
      "deviceAuth":
      {
        "deviceSignature":
        [
          << {1: -7} >>,
          {},
          null,
          h'E99521A85AD75943BF5AEAE68943BF5AE249943BF5AE943BF5AE43BF5AEDE99521A85AD75943BF
          5AEAE
          68943BF5AE249943BF5AE943BF5AE43BF5AED'

        ]
      }
    }
  }
  ],
  "status": 0
}
```

Passo 13: L'Istanza di Relying Party riceve il `SessionData`, quindi DEVE decifrarlo e DEVE verificare la firma dell'Istanza del Wallet per garantire l'integrità dei dati e che provengano dal dispositivo previsto (device binding). DEVE anche controllare la validità dell'mdoc, inclusa la firma del suo Fornitore di Attestati Elettronici. In caso di Attestati Elettronici Digitali di lunga durata, DOVREBBE anche controllare lo stato di revoca utilizzando il TOKEN-STATUS-LIST.

Passo 14: Una volta completato lo scambio di dati, una delle parti può terminare la sessione. La sessione può essere terminata inviando lo status code per la *Session Termination* in un messaggio `SessionData`; questo può essere inviato insieme a una la Richiesta mdoc o Risposta mdoc [ISO18013-5 #12.2.4] (*WP_113c* (pagina 798)). Se viene utilizzato BLE, questo può comportare l'invio di uno *status code* per la *Session Termination* o il comando "End". In questo scenario, il Client GATT (Istanza di Relying Party) DEVE annullare l'iscrizione dalle caratteristiche e disconnettersi dal server GATT (Istanza del Wallet) (PPR-007, *WP_113b* (pagina 798), and *WP_114* (pagina 798)).

Considerazione Finale: Il flusso di presentazione si è concentrato sullo scambio tecnico di dati in contesti di prossimità. È cruciale riconoscere che i flussi di prossimità supervisionati che coinvolgono un verificatore umano svolgono un ruolo vitale in molti casi d'uso (ad esempio, verifica dell'età in un negozio, controllo dell'identità da parte delle forze dell'ordine). L'elemento umano aggiunge un livello di verifica dell'identità attraverso l'ispezione visiva e il confronto, contribuendo agli aspetti di User Binding e garanzia di autenticazione complessiva non completamente catturati in un flusso di presentazione puramente tecnico.

Nota

Durante ciascuna transazione di presentazione di credenziali eseguita tramite il Proximity Flow, la Wallet Instance DEVE creare e mantenere un corrispondente record di transazione nel registro delle transazioni (vedere *Dashboard dell'Istanza del Wallet e Registrazione delle Transazioni* (pagina 319)).

Il record di transazione DEVE essere creato una volta che la Wallet Instance ha stabilito con successo la sessione e ha accettato il reader per l'elaborazione (ossia, dopo aver decifrato il messaggio `SessionEstablishment` e verificato l'autenticazione del reader, Passo 10). A questo punto, il record DEVE includere i metadati della transazione e il contesto della richiesta disponibile in quella fase (ad esempio, il/i tipo/i di attestazione richiesti come `docType` e l'/gli identificativo/i degli attributi richiesti), senza registrare alcun valore degli attributi.

Il record DEVE essere aggiornato man mano che la transazione procede, in modo da riflettere l'evoluzione dello stato della transazione e il contesto del risultato (ad esempio, quanto effettivamente presentato dopo il consenso dell'Utente e la preparazione/invio della risposta, Passi 11–12), senza registrare alcun valore degli attributi.

Il record DEVE essere finalizzato al termine della transazione, indicando l'esito (ad esempio, completata, fallita o sessione terminata; Passi 13–14 e Session Termination).

`DeviceEngagement` tramite QR Code

La figura seguente illustra il flusso di basso livello conforme a ISO 18013-5 per `DeviceEngagement` tramite QR Code corrispondente al Box A nella Figura *Flusso di Presentazione di Basso Livello in prossimità.* (pagina 541).

Passo 1: L'Istanza del Wallet presenta un QR Code all'Istanza di Relying Party. Il QR code DEVE contenere un URI con "mdoc:" come schema e la struttura `DeviceEngagement` specificata nella Sezione 9.1 codificata utilizzando, come percorso, base64url-without-padding, secondo RFC 4648 (*WP_102a* (pagina 798)).

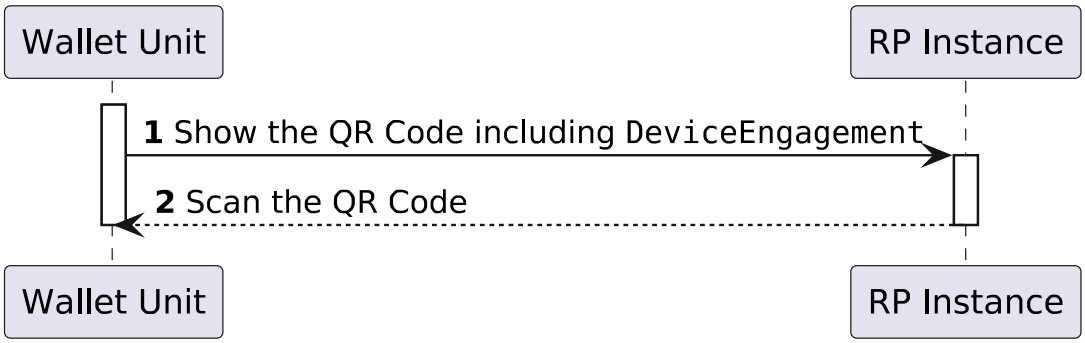


Fig. 12.16: Device Engagement tramite QR Code.

Esempio Non Normativo con BLE come Device Retrieval

Di seguito è riportato un esempio non normativo di"DeviceEngagement" nella notazione diagnostica che utilizza QR per il *Device Engagement* e Bluetooth Low Energy (BLE) per il recupero dei dati.

```
{
  0: "1.1", % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
  ],
  2: % DeviceRetrievalMethods (Device engagement using QR code with BLE for retrieval)
  [
    [
      2, % BLE
      1, % Version
      { % BLE options
        0: false, % no support for mdoc peripheral server mode
        1: true,  % support for mdoc central client mode
        11: h'45EFEF742B2C4837A9A3B0E1D05A6917' % UUID of mdoc client central mode
      }
    ]
  ],
```

```
  5: % OriginInfos (Required because Capabilities is present)
  [],
  6: % Capabilities (Defines supported features)
  {
    2: false, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
    3: true % ReaderAuthAllSupport (Supports reader authentication)
  }
}
```

Esempio Non Normativo con NFC come Device Retrieval

Di seguito è riportato un esempio non normativo di `DeviceEngagement` nella notazione diagnostica che utilizza QR per il *Device Engagement* e NFC per il recupero dei dati.

```
{
  0: "1.1", % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
  ],
  2: % DeviceRetrievalMethods (Device engagement using QR code with BLE for retrieval)
  [
    [
      1, % NFC
      1, % Version
      { % NFC options
        0: uint, % Maximum length of command data field
        1: uint, % Maximum length of response data field
      }
    ]
  ],
  5: % OriginInfos (Required because Capabilities is present)
  [],
  6: % Capabilities (Defines supported features)
  {
    2: false, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
```

```
        3: true % ReaderAuthAllSupport (Supports reader authentication)
      }
    }
```

Passo 2: Il verificatore utilizza la sua Istanza di Relying Party per scansionare il QR code e recuperare i dati `DeviceEngagement` dall'mdoc. Esso DEVE selezionare una delle tecnologie di trasmissione tra quelle fornite nella struttura `DeviceEngagement`.

`DeviceEngagement` tramite NFC

La figura seguente illustra il flusso di basso livello conforme a ISO 18013-5 per `DeviceEngagement` tramite NFC corrispondente al Box A nella Figura *Flusso di Presentazione di Basso Livello in prossimità.* (pagina 541) (*WP_103* (pagina 798)).

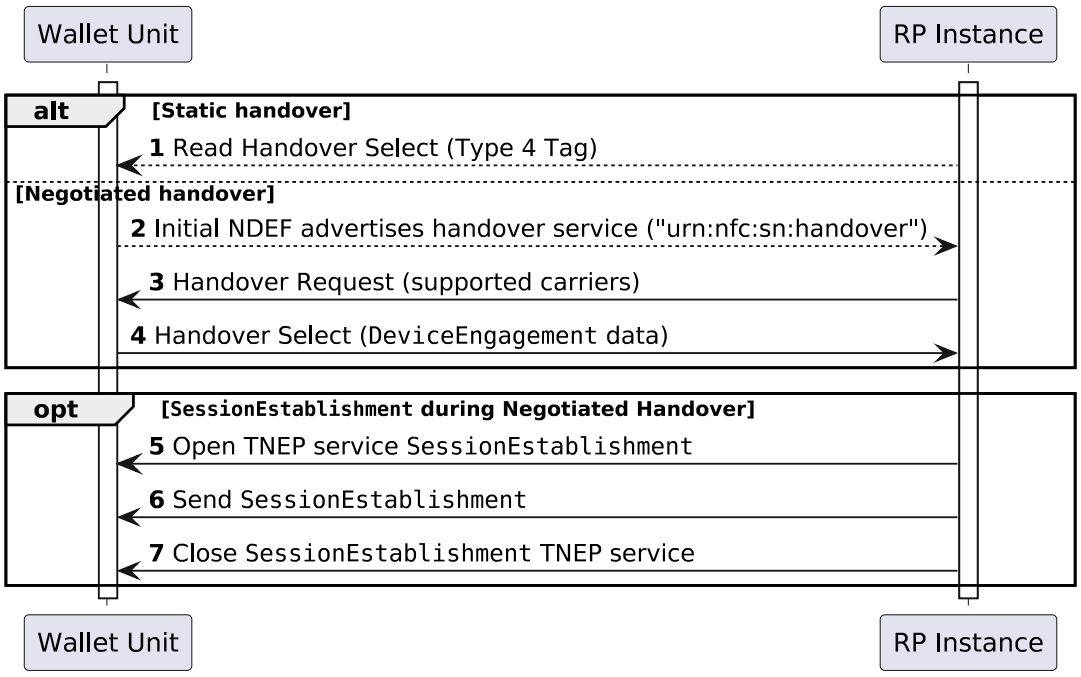


Fig. 12.17: Device Engagement tramite NFC.

`DeviceEngagement` tramite NFC è basato su *NFC Forum Connection Handover Technical Specification*, Versione 1.5. È supportata solo la modalità Reader/Writer utilizzando un Tag di Tipo 4. Il protocollo Connection Handover è sempre avviato dall'Istanza di Relying Party, che assume il ruolo di *Handover Requester*. L'Istanza del Wallet agisce come NFC Tag Device e l'Istanza di Relying Party come NFC Reader Device. L'Istanza del Wallet DEVE utilizzare Static Handover o *Negotiated Handover*: - **Static Handover**: L'Istanza di Relying Party recupera un messaggio *Handover Select* direttamente dal Tag di Tipo 4 dell'Istanza del Wallet. Questo messaggio contiene almeno un Alternative Carrier Record, ognuno indicante un metodo di recupero supportato dall'Istanza del Wallet (*WP_103a* (pagina 798)). L'Istanza di Relying Party DEVE selezionare una di queste tecnologie di trasmissione. (vedere Passo 1) - **Negotiated**

Handover: L'Istanza del Wallet include il servizio `urn:nfc:sn:handover` in un Service Parameter Record del messaggio NDEF (NFC Data Exchange Format) iniziale (*WP_103b* (pagina 798)). Selezionando questo servizio, l'Istanza di Relying Party invia una *Handover Request* con un Alternative Carrier Record per ogni carrier che supporta. L'Istanza del Wallet risponde con un messaggio *Handover Select* contenente esattamente un carrier selezionato. (Vedere Passi 2-4)

Passo 1: L'Istanza di Relying Party legge il Tag NFC di Tipo 4 del Wallet per ottenere un messaggio *Handover Select*, che include: - Alternative Carrier Record: un record NDEF all'interno di un messaggio *Handover Select* o *Handover Request*. Punta a una possibile tecnologia di comunicazione (chiamata "carrier"), come NFC o BLE. Informa il lettore sul carrier supportato e un puntatore (Auxiliary Data Reference) a informazioni più dettagliate. L'Alternative Carrier Record per la tecnologia di trasmissione *Device Retrieval* NFC DEVE fare riferimento al Carrier Configuration Record con il riferimento ID "nfc". - Carrier Configuration Record: fornisce i parametri tecnici necessari per utilizzare effettivamente quel carrier. Per la tecnologia di trasmissione *Device Retrieval NFC*, DEVE avere il tipo "iso.org:18013:nfc" e il riferimento ID "nfc". Il contenuto binario del Carrier Configuration Record DEVE essere codificato secondo la Tabella 1 di [ISO18013-5 #9.2.2] (*WP_103d* (pagina 798)).

Per esempio: Per NFC, questo definisce le lunghezze massime di comando/risposta APDU (Application Protocol Data Unit); Per BLE, definisce l'UUID del servizio dell'Istanza del Wallet, gli UUID delle caratteristiche, la dimensione MTU (Maximum Transmission Unit) e parametri di connessione opzionali; Se è supportato il `SessionEstablishment` anticipato, elenca anche il nome del servizio TNEP (Tag NDEF Exchange Protocol) utilizzato per inviare il messaggio `SessionEstablishment` durante l'handover.

Nota

Per la tecnologia di trasmissione *Device Retrieval NFC*, i contenuti dell'Alternative Carrier Record e del/dei Carrier Configuration Record DEVONO essere conformi a [ISO18013-5 # 9.2.2]. Per la tecnologia di trasmissione *Device Retrieval BLE*, i contenuti dell'Alternative Carrier Record e del/dei Carrier Configuration Record devono essere conformi a [ISO18013-5 #11.1.2].

- Auxiliary Data Record DEVE trasportare la struttura `DeviceEngagement` dall'Istanza del Wallet all'Istanza di Relying Party come parte del record NDEF ausiliario nel messaggio *Handover Select*. Questo record ha il tipo `iso.org:18013:deviceengagement`, il riferimento ID "mdoc", e utilizza il formato di tipo esterno del forum NFC (`0x04`). Per ogni record Alternative Carrier, l'Auxiliary Data Reference DEVE puntare al record NDEF contenente la Struttura `DeviceEngagement` (*WP_103e* (pagina 798)).

Passo 2: L'Istanza di Relying Party legge il messaggio NDEF (NFC Data Exchange Format) Iniziale dell'Istanza del Wallet, che contiene un service parameter record per `urn:nfc:sn:handover`, indicando che il Wallet supporta *Negotiated Handover*.

Passo 3: L'Istanza di Relying Party invia una *Handover Request* all'Istanza del Wallet elencando i carrier supportati.

Passo 4: L'Istanza del Wallet restituisce *Handover Select* costruito in risposta al messaggio *Handover Request* ricevuto. I contenuti del messaggio *Handover Select* sono gli stessi del Passo 1 (*WP_103f* (pagina 798)).

Nota

L'uso di *Negotiated Handover* per il *Device Engagement* consente la negoziazione dei metodi di trasferimento. Per BLE, consente inoltre la negoziazione delle chiavi utilizzate dal livello di trasmissione. Questo fornisce un'esperienza utente migliorata e migliora la sicurezza della trasmissione dei dati [ISO18013-5 #9.2.1].

Nota

Procedere solo se le Capabilities di `DeviceEngagement` includono `HandoverSession EstablishmentSupport` impostato su `true` (*WP_103c* (pagina 798)). Altrimenti, saltare il `SessionEstablishment` anticipato. Il `SessionEstablishment` anticipato viene inviato tramite un servizio TNEP dedicato; lo stesso `SessionEstablishment` DEVE anche essere inviato nuovamente durante il *Device Retrieval* e DEVE corrispondere. Se non corrisponde, l'Istanza del Wallet termina (*WP_103g* (pagina 798)). Se il `SessionEstablishment` anticipato non riesce a essere inviato, procedere normalmente (*WP_103h* (pagina 798)).

Passo 5: [Opzionale] L'Istanza di Relying Party apre il servizio TNEP denominato [urn:placeholder] con l'Istanza del Wallet durante l'handover negoziato per consegnare il messaggio `SessionEstablishment` anticipato.

Passo 6: L'Istanza di Relying Party invia `SessionEstablishment` (ad esempio, `EReaderKey` + `DeviceRequest` cifrato). L'Istanza del Wallet lo elabora; il *Device Retrieval* non è ancora iniziato.

Passo 7: L'Istanza di Relying Party chiude il servizio TNEP.

Nota

Se un messaggio `SessionEstablishment` opzionale viene inviato durante *Negotiated Handover* (Passo 5), l'Istanza del Wallet DEVE verificare che corrisponda al messaggio `Session Establishment` ricevuto durante *Device Retrieval* (utilizzando BLE o canale sicuro NFC). Questa verifica è richiesta per garantire un corretto Session Binding.

Esempio Non Normativo

Di seguito è riportato un esempio non normativo di una struttura `DeviceEngagement` per *Device Retrieval* tramite BLE e NFC.

```
{
  0: "1.1", % Version (Updated to 1.1 because Capabilities and OriginInfos are present)
  1: % Security
  [
    1, % defines the cipher suite , which contains only EC curves
    24(<< % embedded CBOR data item
    {
      1: 2, % kty:EC2 (Elliptic curves with x and y coordinate pairs)
      -1: 1, % crv:p256
      -2:h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE', %
      x-coordinate
      -3:h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67' %
      y-coordinate
    }
    >>)
  ],
  2: % DeviceRetrievalMethods (Absent when using NFC for device engagement)
  [],
  5: % OriginInfos (Required because Capabilities is present)
  [],
  6: % Capabilities (Defines supported features)
  {
    2: true, % HandoverSessionEstablishmentSupport (Supports negotiated handover)
    3: true % ReaderAuthAllSupport (Supports reader authentication)
  }
}
```

`SessionEstablishment` tramite BLE

La figura seguente illustra il flusso di basso livello conforme a ISO18013-5 per `Session Establishment` tramite BLE corrispondente ai Box B nella Figura 8.10.

Passo 1: L'Istanza del Wallet e l'Istanza di Relying Party stabiliscono una connessione BLE sicura [ISO18013-5 #11.1]. L'Istanza di Relying Party (central) si connette all'Istanza del Wallet (peripheral) utilizzando l'UUID del servizio dell'Istanza del Wallet fornito da DeviceEngagement, individua servizi/caratteristiche e abilita le notifiche secondo necessità (*WP_112c* (pagina 798)).

Passi 2-5: [Opzionale] L'Istanza del Wallet avvia la verifica preparandosi a controllare l'identità della Relying Party tramite la caratteristica Ident, che è una caratteristica BLE GATT che trasporta un valore identificatore come descritto in [ISO18013-5 #11.1.3.2]. L'Istanza del Wallet deriva il valore Ident atteso e legge la caratteristica Ident della Relying Party, confrontandola

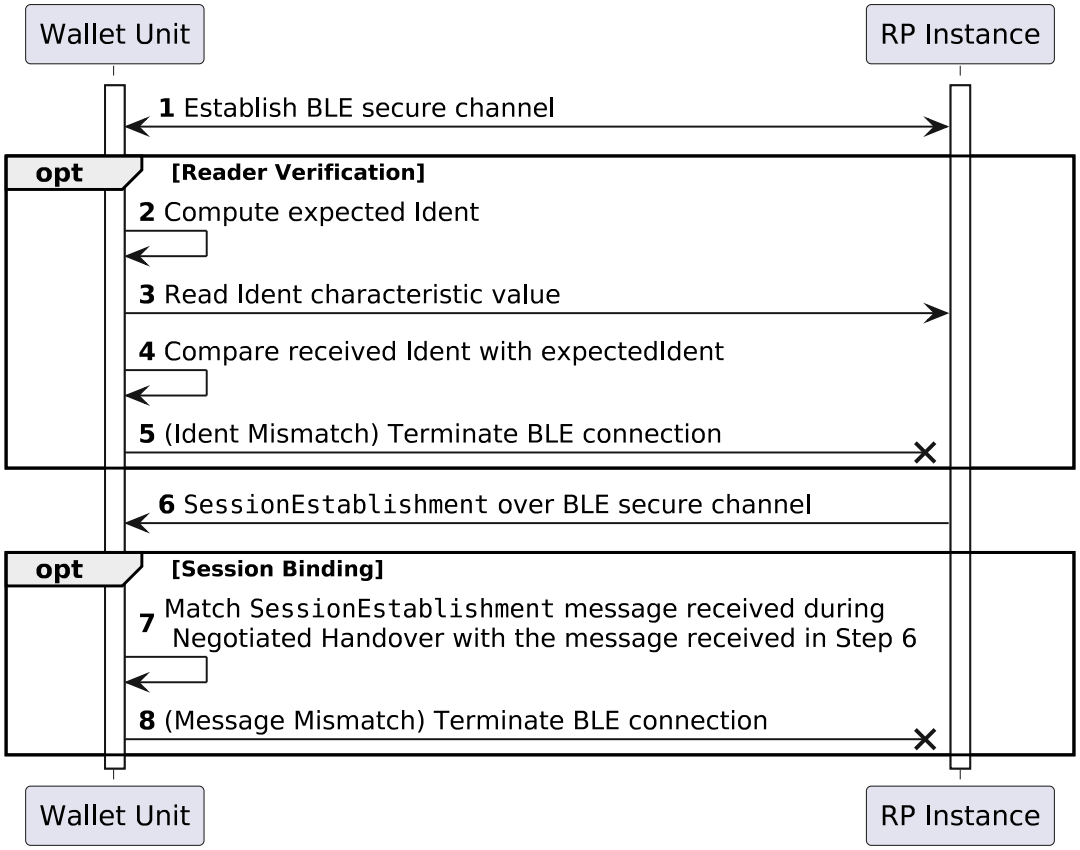


Fig. 12.18: Session Establishment tramite BLE.

con l'Ident atteso, e terminando la connessione BLE qualora non vi sia corrispondenza (*WP_112d* (pagina 798)).

Nota

Lo scopo della caratteristica Ident è unicamente di verificare se l'Istanza del Wallet è connessa all'Istanza di Relying Party corretta prima di iniziare il *Device Retrieval*. Se l'Istanza del Wallet è connessa all'Istanza di Relying Party sbagliata, il *Session Establishment* fallirà. Connettersi e disconnettersi a un'Istanza di Relying Party richiede una quantità relativamente grande di tempo; pertanto, è più veloce implementare metodi per identificare l'Istanza di Relying Party corretta [ISO18013-5 #11.1.3.1].

Passo 6: L'Istanza di Relying Party invia il messaggio `SessionEstablishment` cifrato (includendo `EReaderKey` e `DeviceRequest` cifrato) tramite la connessione BLE stabilita.

Passi 7-8: [Opzionale] Se l'Istanza del Wallet riceve il messaggio `SessionEstablishment` durante *Negotiated Handover*, l'Istanza del Wallet DEVE verificare se questo messaggio `SessionEstablishment` corrisponde al messaggio `SessionEstablishment` ricevuto durante la fase di *Device Retrieval* (cioè, Passo 6). In caso di mancata corrispondenza, l'Istanza del Wallet deve terminare la connessione BLE [ISO18013-5 #9.2.3].

`SessionData` tramite BLE

La figura seguente illustra il flusso di basso livello conforme a ISO18013-5 per `SessionData` tramite BLE corrispondente ai Box C nella Figura 8.10.

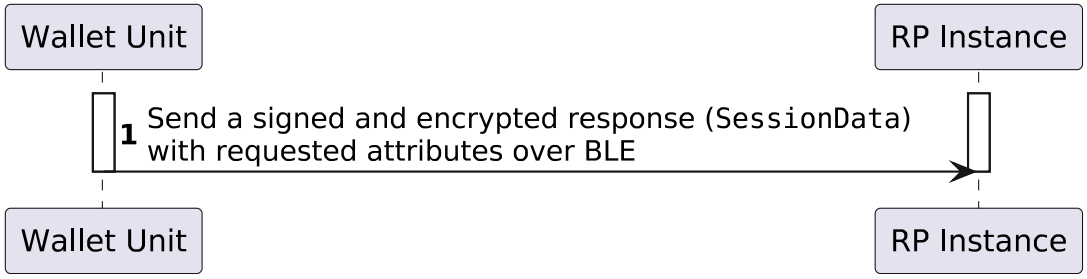


Fig. 12.19: Session Data tramite BLE.

Passo 1: L'Istanza del Wallet invia l'APDU finale contenente l'ultimo blocco DeviceResponse (con attributi richiesti) o uno status code, dopo il quale la sessione può terminare o continuare con una nuova richiesta.

`SessionEstablishment` tramite NFC

Nota

Se il *Device Engagement* viene avviato tramite un QR code, l'Istanza di Relying Party non ha un modo standardizzato per segnalare la sua intenzione di utilizzare NFC per il successivo trasferimento di dati. Questo potrebbe portare a una peggiore esperienza utente, poiché l'Utente potrebbe non essere consapevole di dover utilizzare NFC. Questo problema viene evitato quando NFC viene utilizzato per il *Device Engagement*, poiché stabilisce implicitamente il metodo di trasferimento dati [ISO18013-5 #8.2.5].

Nota

A causa della velocità di trasferimento dati limitata di NFC, se è richiesta una grande quantità di dati per una transazione, potrebbe non essere né pratico né ragionevole per un Utente mantenere il dispositivo all'interno del campo RF dell'Istanza di Relying Party per la durata della transazione. Inoltre, quando un dispositivo lascia il campo RF il segnale viene perso necessitando l'avvio di una nuova transazione. Questo può essere evitato facendo sì che tutte le interazioni dell'Utente con l'Istanza del Wallet vengano effettuate mentre l'Istanza del Wallet rimane nel campo RF oppure, se non sono richieste interazioni dell'Utente durante la fase di trasmissione [ISO18013-5 #8.2.5].

La figura seguente illustra il flusso di basso livello conforme a ISO18013-5 per `Session Establishment` tramite NFC corrispondente al Box B nella Figura 8.10.

Definizioni (Acronimi e Comandi)

Acronimo/ Comando	Descrizione
PICC	Proximity Integrated Circuit Card, implementata dalle Istanze di Wallet che agiscono come tag NFC.
PCD	Proximity Coupling Device, implementato dalle Istanze di Relying Party utilizzando scambi *APDU*.
AID	Application Identifier, ID univoco utilizzato nel comando SELECT *APDU* per coinvolgere l'Istanza del Wallet.
APDU	Application Protocol Data Unit, un formato di messaggio standard per la comunicazione tra Istanza di Relying Party (PCD) e Wallet (PICC) costituito da Command APDU (ad esempio, *SELECT*, *ENVELOPE*, *GET RESPONSE*) dal lettore e Response *APDU* dal wallet, che includono scambio di dati e fine scambio utilizzando status words (*SW1/ SW2*).

continues on next page

Tabella 12.31 – continua dalla pagina precedente

Acronimo/ Comando	Descrizione
SELECT APDU	Comando che apre l'Istanza del Wallet tramite *AID*. La risposta include File Control Information (FCI) e Status Word (*SW1/SW2*).
ENVELOPE APDU	Comando che trasporta messaggi di sessione (ad esempio, `SessionEstablishment`). La risposta indica lo stato di elaborazione (*OK* o *more data*).
GET RESPONSE APDU	Comando che recupera dati di risposta aggiuntivi quando il Wallet segnala che "più dati" sono disponibili (*SW1=61*).
SW1/SW2	*SW1/SW2* (Status Words) — status code a due byte alla fine di ogni risposta. Valori comuni: *90 00 = successo*, *61 XX = more data*, *6A 82 = file/applicazione non trovata*.

Passo 1: L'Istanza di Relying Party (PCD) invia un comando SELECT *APDU* con l'Application Identifier (*AID: A0 00 00 02 48 04 00*) per coinvolgere l'Istanza del Wallet.

Passo 2: L'Istanza del Wallet (PICC) risponde con File Control Information (FCI) e status words (SW1/SW2), confermando il successo (*90 00*) o indicando che ci sono più dati (*61 XX*) (*WP_112e* (pagina 798)).

Passo 3: L'Istanza di Relying Party invia un ENVELOPE *APDU* che trasporta il messaggio `SessionEstablishment`, che contiene il `DeviceRequest` cifrato e la sua chiave pubblica effimera per la configurazione della sessione.

Passo 4: L'Istanza del Wallet elabora `SessionEstablishment` e restituisce una risposta *APDU* con *SW1/SW2* (*OK* o *more data* da recuperare), confermando l'inizio del contesto di sessione sicura (*WP_112f* (pagina 798)).

Passi 5-6: [Opzionale] L'Istanza del Wallet riceve il messaggio `SessionEstablishment` durante *Negotiated Handover*, l'Istanza del Wallet DEVE verificare che questo messaggio `SessionEstablishment` corrisponda allo stesso messaggio ricevuto durante la fase di *Device Retrieval* (cioè, Passi 3-4). In caso di mancata corrispondenza, l'Istanza del Wallet DEVE terminare la connessione NFC [ISO18013-5 #9.2.3].

`SessionData` tramite NFC

La figura seguente illustra il flusso di basso livello conforme a ISO18013-5 per `SessionData` tramite NFC corrispondente al Box C nella Figura 8.10.

Passi 1-2: Finché l'Istanza del Wallet segnala che più dati sono disponibili (*61 XX*), l'Istanza di Relying Party emette *GET RESPONSE APDU* per richiedere il blocco successivo. L'Istanza del Wallet restituisce frammenti `SessionData` cifrati fino a quando tutti i dati sono consegnati.

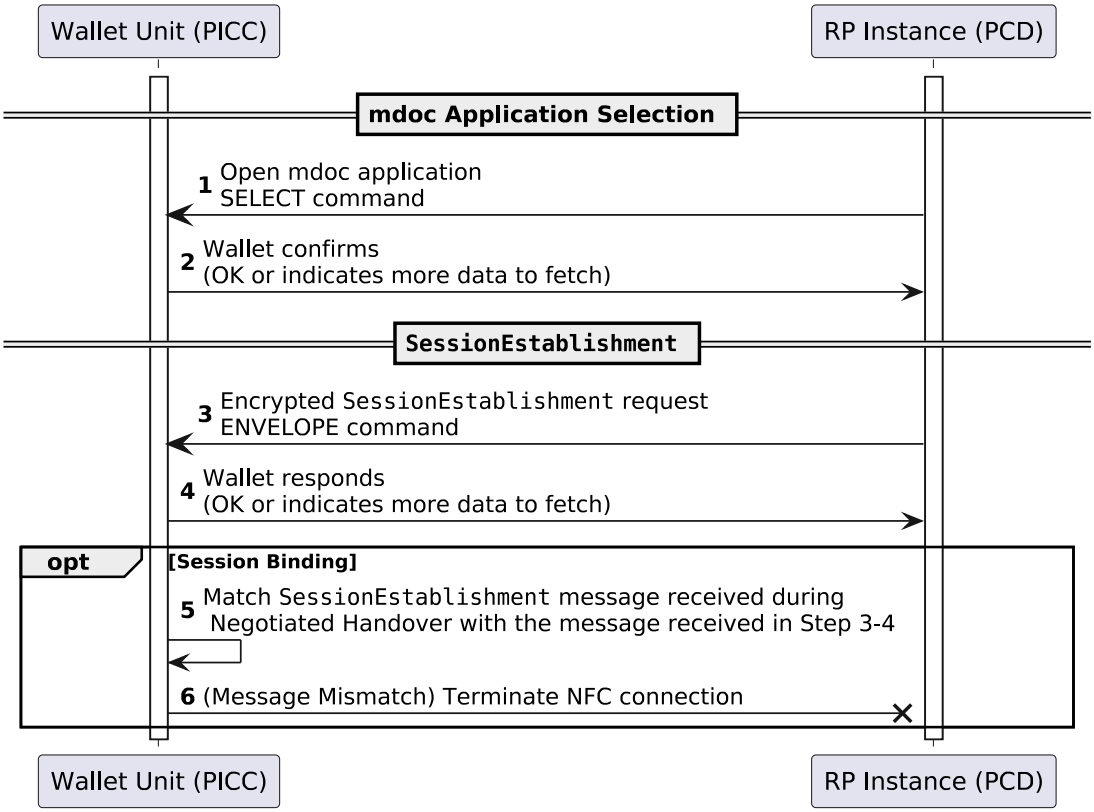


Fig. 12.20: Session Establishment tramite NFC.

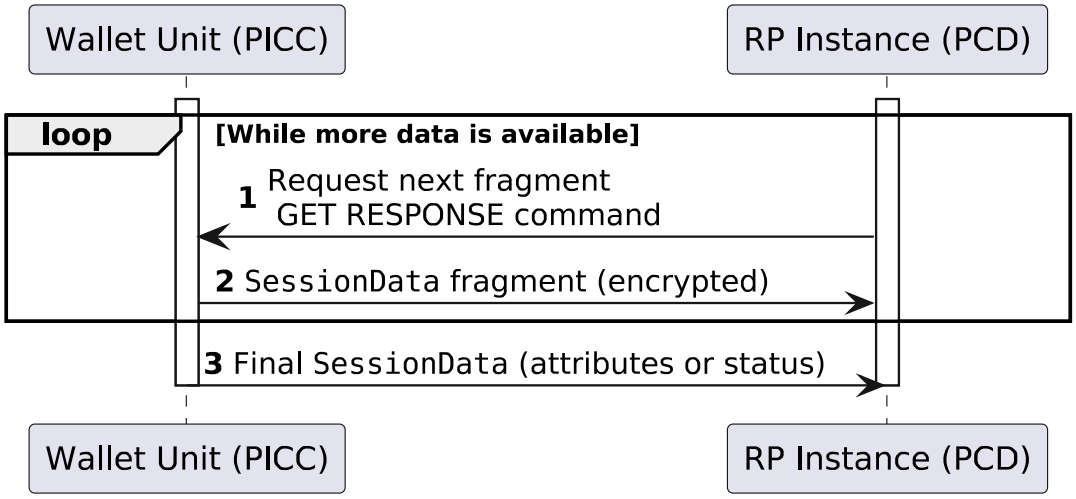


Fig. 12.21: Session Data tramite NFC.

Passo 3: L'Istanza del Wallet invia l'*APDU* finale contenente l'ultimo blocco DeviceResponse (con attributi richiesti) o un status code, dopo il quale la sessione può terminare o continuare con una nuova richiesta.

Device Engagement

La struttura del Device Engagement DEVE essere codificata in CBOR e avere almeno le seguenti componenti (PPR-001, PPR-021, PPR-022, and *WP_102* (pagina 798)):

Componente	Descrizione
Version	*(tstr)*. Versione della struttura *Device Engagement*.
Security	*(array)*. Contiene due valori obbligatori: • *(int)*. Identificatore della suite di cifratura. Vedere Tabella 22 di ISO18013-5. • *(bstr)*. Chiave pubblica effimera generata dall'Istanza del Wallet, utilizzata dall'Istanza di Relying Party per derivare la chiave di sessione. La chiave DEVE essere di un tipo consentito dalla suite di cifratura selezionata (PPR-022).
DeviceRetrieval Mode-BLEOptions	*(map)*. Fornisce opzioni per la connessione BLE, come modalità Peripheral Server o Central Client, e l'UUID del dispositivo. Vedere Tabella 2 di ISO18013-5 per la mappatura dettagliata. Se l'Istanza del Wallet indica durante il *Device Engagement* che supporta entrambe le modalità, l'Istanza di Relying Party DOVREBBE selezionare la modalità mdoc central client [ISO18013-5 #11.1.3.1]. Presente solo quando si esegue *Device Engagement* utilizzando il QR code. Assente quando si utilizza NFC per eseguire *Device Engagement*.
DeviceRetrieval Mode-NFCOptions	*(map)*. Fornisce opzioni per le connessioni NFC, incluso il ruolo supportato (PICC o PCD) e le dimensioni massime di comando/risposta PDU. Vedere Tabella 2 di ISO18013-5 per la mappatura dettagliata. Nel caso in cui NFC venga utilizzato per *Device Retrieval*, l'Istanza del Wallet DEVE supportare la modalità PICC e l'Istanza di Relying Party DEVE supportare la modalità PCD [ISO18013-5 #11.2]. Quest'ultima è presente solo quando si esegue *Device Engagement* utilizzando il QR code, mentre è assente quando si utilizza NFC per eseguire *Device Engagement*.

continues on next page

Tabella 12.32 – continua dalla pagina precedente

Componente	Descrizione
Capabilities	*(map)*. Dichiara le capacità opzionali supportate dall'mdoc, che sono: • **HandoverSessionEstablishmentSupport** *(bool)*. Se presente, DEVE essere impostato su *true*. Indica il supporto per ricevere il messaggio *SessionEstablishment* durante *Negotiated Handover*, come definito in [ISO18013-5 #9.2.3] (PPR-024). • **ReaderAuthAllSupport** *(bool)*. Se presente, DEVE essere impostato su *true*. Indica il supporto per ricevere la struttura *ReaderAuthAll* nella la Richiesta mdoc, come definito in [ISO18013-5 #10.2.6] (PPR-025).
OriginInfos	*(array)*. Descrive l'interfaccia utilizzata per ricevere e consegnare la struttura di engagement. *OriginInfos* PUÒ essere un array vuoto.

Richiesta mdoc

I messaggi nella Richiesta mdoc DEVONO essere codificati utilizzando CBOR. La stringa di byte CBOR risultante per la Richiesta mdoc DEVE essere cifrata con la Chiave di Sessione ottenuta dopo la fase di Device Engagement e DEVE essere trasmessa utilizzando il protocollo BLE o NFC (PPR-026, PPR-027, PPR-028).

Ogni Richiesta mdoc DEVE essere conforme alla seguente struttura e DEVE includere i seguenti componenti, a meno che non sia specificato diversamente:

Componente	Descrizione
version	*(tstr)*. Versione della struttura della Richiesta mdoc. Abilita la gestione della compatibilità tra diverse versioni o profili di implementazione.

continues on next page

Tabella 12.33 – continua dalla pagina precedente

Componente	Descrizione
docRequests	*(array)*. Ogni voce è una *DocRequest* contenente: • **itemsRequest**. Struttura *ItemsRequest* codificata CBOR, formattata come: – **docType** *(tstr)*. Il tipo di documento richiesto. Vedere *Formato Attestato Elettronico mdoc-CBOR* (pagina 389). – **nameSpaces** *(map)*. Una mappa di identificatori di namespace a *DataElements* richiesti. Ogni voce in *DataElements* include: ∗ **DataElementIdentifier** *(tstr)*. L'identificatore dell'elemento dati richiesto. ∗ **IntentToRetain** *(bool)*. Indica se la Relying Party intende conservare il valore dell'elemento dati. • **readerAuth** *(COSE_Sign1, CONDIZIONALE)*. Utilizzato per autenticare l'Istanza di Relying Party per ogni *DocRequest*. La firma è calcolata sui dati *ReaderAuthentication*, come definito in [ISO18013-5 #12.5]. Questo componente DEVE essere presente solo se `readerAuthAll` non viene utilizzato (PPR-025).
readerAuthAll	*(COSE_Sign1, CONDIZIONALE)*. Utilizzato per autenticare la Relying Party una volta per tutte le *DocRequest*. La firma è calcolata sui dati *ReaderAuthenticationAll*, come definito in [ISO18013-5 #12.5]. Questo componente DEVE essere presente solo se `ReaderAuthAllSupport` è impostato su `true` nel messaggio di DeviceEngagement, e i campi individuali `readerAuth` non vengono utilizzati (PPR-025).

Risposta mdoc

I messaggi nella Risposta mdoc DEVONO essere codificati utilizzando CBOR e DEVONO essere cifrati con la Chiave di Sessione ottenuta dopo la fase di Device Engagement (PPR-029, PPR-030).

Ogni Risposta mdoc DEVE essere conforme alla seguente struttura e DEVE includere i seguenti componenti, a meno che non sia specificato diversamente:

Componente	Descrizione
version	*(tstr)*. Versione della struttura Risposta mdoc. Abilita il tracciamento delle modifiche e il mantenimento della compatibilità tra versioni dello standard o profili di implementazione.
documents	*(array of Documents, OPZIONALE)*. Collezione codificata CBOR di documenti restituiti in risposta alla richiesta. Ogni documento include componenti *issuerSigned* e *deviceSigned*, e segue la struttura definita in [ISO18013-5 #10.3.3].
documentErrors	*(map, OPZIONALE)*. Una mappa di codici di errore per documenti non restituiti, come definito in [ISO18013-5 #10.3.6]. Ogni chiave è un *docType*, e ogni valore è un *ErrorCode* (int) che indica il motivo per cui il documento non è stato restituito.
status	*(uint)*. Status code che indica l'esito della richiesta. Ad esempio, *"status": 0* significa elaborazione riuscita. Per dettagli, vedere Tabella 3 (Response Status) di [ISO18013-5 #10.3.5].

Ogni elemento in **documents** DEVE essere conforme alla seguente struttura e DEVE includere i seguenti componenti, a meno che non sia specificato diversamente (PPR-029):

Componente	Descrizione
docType	*(tstr)*. identificativo del tipo di documento. Ad esempio, per un mDL, il valore DEVE essere `org.iso.18013.5.1.mDL`.
issuerSigned	*(bstr)*. Contiene la struttura *IssuerNameSpaces*, che include elementi dati firmati dal Fornitore di Attestati Elettronici, e la struttura *issuer Auth*, che garantisce la loro autenticità e integrità utilizzando il Mobile Security Object (MSO). Vedere *Formato Attestato Elettronico mdoc-CBOR* (pagina 389).
deviceSigned	*(bstr)*. Contiene la struttura *DeviceNameSpaces* (elementi dati firmati dall'Istanza del Wallet), e la struttura *deviceAuth*, che include i dati di autenticazione firmati dall'Istanza del Wallet. Vedere la tabella sottostante per dettagli.
errors	*(map, OPZIONALE)*. Una mappa di codici di errore per ogni elemento dati non restituito raggruppato per namespace. Ogni chiave rappresenta un namespace, e ogni valore è una mappa di identificatori di elementi dati ai corrispondenti codici di errore. Vedere [ISO18013-5 #10.3.6] per dettagli sulla struttura degli errori.

Una struttura di dati **deviceSigned** DEVE essere conforme alla seguente struttura e DEVE includere i seguenti componenti (*WP_111a* (pagina 798)):

Componente	Descrizione
nameSpaces	*(bstr)*. Contiene la struttura *DeviceNameSpaces*. PUÒ essere una struttura vuota. *DeviceNameSpaces* mappa identificatori di namespace verso un insieme di elementi dati firmati dall'Istanza del Wallet. Ogni namespace contiene uno o più *DeviceSignedItem*, dove ogni elemento include: • **DataItemName** *(tstr)*. L'identificatore dell'elemento dati. • **DataItemValue** *(any)*. Il valore dell'elemento dati.
deviceAuth	*(COSE_Sign1)*. Contiene la struttura `DeviceAuth`, che DEVE includere la **deviceSignature** per l'autenticazione dell'Istanza del Wallet. La firma è calcolata sui dati `DeviceAuthentication`, che lega gli elementi restituiti alla sessione e alla richiesta. Vedi [ISO18013-5 #12.4] per i dettagli sulla struttura di autenticazione.

Session Termination

La sessione DEVE chiudersi qualora si verifichi almeno una delle seguenti condizioni (PPR-007 and *WP_113–113a* (pagina 798)):

- Dopo un timeout di inattività nel quale non sono scambiati messaggi di *Session Establishment* o *Session Data*. Questo timeout di inattività viene implementato dall'Istanza del Wallet e dall'Istanza di Relying Party e DOVREBBE essere non inferiore a 300 secondi;
- Quando l'Istanza del Wallet non accetta più richieste;
- Quando l'Istanza di Relying Party non invia ulteriori richieste.

Se l'Istanza del Wallet e l'Istanza di Relying Party non inviano o ricevono ulteriori richieste, la terminazione della sessione DEVE essere avviata come segue (PPR-007 and *WP_113* (pagina 798)):

- Inviare lo status code per la *Session Termination*, o
- Inviare il comando "End" come delineato in [ISO18013-5 #11.1.3.3].

Quando una sessione viene terminata, l'Istanza del Wallet e l'Istanza di Relying Party DEVONO eseguire almeno le seguenti azioni (*WP_114* (pagina 798)):

- Distruzione delle chiavi di sessione e del materiale di chiavi effimere correlato;
- Chiusura del canale di comunicazione utilizzato per il *Device Retrieval*.

Nota

Vedere *Formato Attestato Elettronico mdoc-CBOR* (pagina 389) per il significato degli acronimi di tipo CBOR.

12.2.3 Identity Matching e Identity Reconciliation

L'**identity matching** è il processo con cui una Relying Party accerta che gli attributi di identificazione personale presentati in una transazione si riferiscano alla stessa persona fisica. È lo stesso approccio pratico già utilizzato in qualsiasi infrastruttura legacy di autenticazione e autorizzazione. Non è una funzionalità dell'Istanza del Wallet, né un passo del protocollo di presentazione. La verifica crittografica dell'Attestato Elettronico resta un prerequisito distinto, e l'identity reconciliation è un passo successivo e opzionale.

Dopo la verifica crittografica di un Attestato Elettronico presentato, nei flussi remoto o di prossimità, la Relying Party DEVE prima effettuare l'**identity matching**, per accertare che gli attributi di identificazione personale presentati nella transazione corrente si riferiscano alla stessa persona fisica. Solo dopo un identity matching andato a buon fine, la Relying Party PUÒ effettuare l'**identity reconciliation**, collegando quella persona fisica a una precedente sessione Utente o a un record Utente memorizzato.

Se l'identity matching fallisce, la Relying Party NON DEVE effettuare l'identity reconciliation e NON DEVE trattare chi presenta l'Attestato Elettronico come un Utente già noto.

Questa Sezione specifica le regole di matching e reconciliation che si applicano agli Attestati Elettronici di identificazione personale (PID e IT-Wallet ID). La correlazione delle identità durante l'emissione dell'IT-Wallet ID (IdP–MRTD) è specificata in *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482) ed è fuori dallo scopo di questa Sezione.

Identity Matching

L'identity matching determina se gli attributi di identificazione personale presentati identificano una sola persona fisica.

La Relying Party DEVE applicare il pattern di matching del tipo di Attestato Elettronico presentato, come definito di seguito. Tipi diversi di Attestato Elettronico POSSONO definire pattern di identity matching differenti. Per gli attributi Utente del PID, vedere *Modello di Dati del PID* (pagina 404). Per gli attributi Utente dell'IT-Wallet ID, vedere *Modello di Dati dell'IT-Wallet ID* (pagina 416).

Identity Reconciliation

L'identity reconciliation è il passo successivo che associa una persona fisica identificata con successo a un record Utente o a una sessione della Relying Party.

La Relying Party NON DEVE completare l'identity reconciliation se l'identity matching non è andato a buon fine. La reconciliation PUÒ creare un nuovo record Utente quando nessun record memorizzato corrisponde.

Pattern di Matching per Tipo di Attestato Elettronico

Unique National Identifier Binding

Quando è presentato `personal_administrative_number` o `tax_id_code`, la Relying Party DEVE utilizzarlo come chiave primaria di identity matching prima di qualsiasi identity reconciliation con un record Utente memorizzato (*CI_201* (pagina 825)).

Per l'IT-Wallet ID è presente almeno uno tra `tax_id_code` e `personal_administrative_number`. La Relying Party DEVE utilizzare l'identificativo nazionale univoco presentato come chiave primaria di matching.

Attribute-Based Binding

Per il PID, `personal_administrative_number` è OPZIONALE. Le Relying Party NON DEVONO assumere che un identificativo nazionale univoco sia sempre disponibile.

Quando `personal_administrative_number` non è presentato, la Relying Party DEVE effettuare *Attribute-Based Binding* confrontando almeno `family_name`, `given_name`, `birth_date` (o `birthdate`) e `place_of_birth` dopo normalizzazione di maiuscole/minuscole, spazi e diacritici. La Relying Party DEVE applicare la stessa normalizzazione agli attributi presentati e al record Utente memorizzato, inclusi i membri stringa di `place_of_birth` (`country`, `region`, `locality`). Il confronto della data di nascita DEVE usare la data di calendario ISO 8601 (`YYYY-MM-DD`), indipendentemente dal formato di visualizzazione.

La Relying Party NON DEVE completare l'identity matching su un sottoinsieme di tali attributi che consentirebbe uno scambio di persona (*CI_202* (pagina 825)). In particolare:

- il matching della sola coppia di nomi, del solo nome e della data di nascita, o di qualsiasi sottoinsieme che ometta `place_of_birth`, NON DEVE essere trattato come identity matching andato a buon fine;
- una discordanza su qualsiasi attributo del set minimo DEVE far fallire l'identity matching; la Relying Party NON DEVE ignorare una discordanza sul nome attuale (ad esempio dopo un cambio di nome) per forzare un match;

- se l'Attribute-Based Binding corrisponde a più di un record Utente memorizzato, la Relying Party NON DEVE completare l'identity matching. PUÒ richiedere un identificativo nazionale univoco, oppure applicare un metodo di binding supplementare che identifichi univocamente la persona.

Metodi di Binding Supplementari

I metodi seguenti POSSONO integrare l'Attribute-Based Binding. NON DEVONO sostituire il set minimo di Attribute-Based Binding quando non è presentato un identificativo nazionale univoco:

- **Session-Based Binding**: la presentazione avviene all'interno di una sessione autenticata già associata all'Utente;
- **Issuer-Attested Binding**: l'Issuer attesta che gli attributi presentati appartengono allo stesso soggetto (ad esempio tramite un identificativo di soggetto stabile dell'Issuer);
- **Identificativi specifici della Relying Party**: identificativi precedentemente emessi dalla Relying Party a quell'Utente dopo un identity matching andato a buon fine;
- **Cryptographic Binding**: dimostrando che le chiavi private degli Attestati Elettronici presentati sono gestite dallo stesso Keystore o WSCD.

Privacy e Selective Disclosure

La Selective Disclosure DEVE essere utilizzata affinché gli attributi identificativi necessari solo per il matching non siano rilasciati quando non richiesti dalla specifica transazione, in linea con l'Articolo 5a del Regolamento sull'Identità Digitale Europea e con i principi di combined presentation privacy-preserving dell'ARF, da considerarsi pienamente applicabili quando IT-Wallet sarà notificato come soluzione EUDIW.

Applicazione all'Eliminazione degli Attributi dell'Utente

Quando l'Utente richiede la cancellazione degli attributi presentati, la Relying Party DEVE individuare univocamente uno o più Attestati Elettronici applicando l'identity matching, come specificato in *Eliminazione degli Attributi dell'Utente* (pagina 316).

Quando l'Utente si autentica all'Endpoint di Cancellazione presentando un Attestato Elettronico dall'Istanza del Wallet, la Relying Party DEVE applicare i pattern di matching di questa Sezione.

Quando l'Utente si autentica all'Endpoint di Cancellazione senza usare l'Istanza del Wallet, la Relying Party PUÒ effettuare l'identity matching e l'identity reconciliation utilizzando gli attributi di identificazione personale ottenuti da uno schema nazionale di autenticazione

preesistente (ad esempio CieID), ove tali attributi identifichino univocamente l'Utente nei record della Relying Party.

12.3 Flusso di Recupero dei Wallet Metadata

Alcuni flussi relativi agli Attestati Elettronici richiedono il recupero delle informazioni del Wallet prima di interagire con il Wallet stesso. Ad esempio, per abilitare l'opzione di selezione della Soluzione Wallet durante un flusso di Credential Offer o un flusso di presentazione. A tal fine, le terze parti che offrono il Credential Offer (ad esempio, Authentic Source, Registro, Catalogo) e le Relying Party consultano il Registro del Sistema IT-Wallet, in particolare il Registro di Federazione, tramite gli endpoint del Trust Anchor.

In primo luogo, con l'Immediate Subordinate Listing endpoint ottengono la lista di tutti i subordinati del Trust Anchor identificati dal loro `entity_id`. Per ottenere solo l'`entity_id` relativi ai Wallet Provider, è POSSIBILE utilizzare il parametro di query `entity_type` con valore `openid_wallet_provider`. Se il Trust Anchor non supporta questa funzione, DEVE utilizzare il codice di stato HTTP 400 e il Content Type `application/json`, con il codice di errore `unsupported_parameter`.

Quindi, dato l'`entity_id`, per ciascun Wallet Provider recuperano la corrispondente configurazione utilizzando l'endpoint dei metadati della federazione (GET .well-known/openid-federation) e ottengono i metadati della Wallet Solution (vedere *Metadati della Soluzione Wallet* (pagina 329)). Maggiori dettagli sull'uso degli endpoint del Trust Anchor sono forniti in *L'Infrastruttura di Trust* (pagina 117).

All'interno dei metadati, le terze parti che supportano il Credential Offer e le Relying Party recuperano le informazioni necessarie per la Selection Page, ovvero il logo e il nome completo delle Soluzioni Wallet. Maggiori dettagli sulla progettazione della Selection Page sono disponibili in *Design dell'Esperienza Utente* (pagina 39).

CAPITOLO 13

Endpoints

13.1 Endpoint del Fornitore di Wallet

Il Fornitore di Wallet, responsabile della fornitura di una Soluzione Wallet, DEVE esporre gli endpoint per supportare l'instaurazione della fiducia e le funzionalità essenziali dell'Istanza di Wallet. Questi includono l'endpoint di Federazione `/.well-known/openid-federation` che DEVE aderire alla specifica OpenID Federation 1.0 per stabilire in modo affidabile la fiducia con il Fornitore di Wallet, nonché endpoint per la registrazione dell'Istanza di Wallet, la generazione di nonce (richiesta per la registrazione), l'emissione di attestati e la revoca. A parte l'endpoint di Federazione, i dettagli di implementazione degli altri sono lasciati alla discrezione del Fornitore di Wallet.

Nota

I test relativi all'uso degli endpoint del Wallet Provider sono definiti in *Matrice di Test per Wallet Provider* (pagina 778), in particolare in *Casi di Test per Backend del Fornitore del Wallet* (pagina 778), *Casi di Test per Istanza del Wallet* (pagina 781) e *Casi di Test Opzionali per Istanza del Wallet* (pagina 814).

13.1.1 Endpoint di Federazione

L'endpoint `/.well-known/openid-federation` serve come meccanismo di discovery per l'instaurazione della fiducia recuperando la Entity Configuration del Fornitore di Wallet.

Vedere la Sezione *Entity Configuration del Fornitore di Wallet* (pagina 326) per i dettagli tecnici (*WP_001–004* (pagina 778)).

13.1.2 Endpoint Nonce della Soluzione Wallet

Questo è un endpoint API RESTful che consente all'Istanza di Wallet di richiedere un nonce crittografico dal Fornitore di Wallet. Il nonce serve come sfida imprevedibile, monouso per garantire la freschezza e prevenire attacchi di replay.

Vedere *Richiesta di Nonce dell'Applicazione Mobile* (pagina 710) e *Risposta di Nonce dell'Applicazione Mobile* (pagina 711) per i dettagli sulla Richiesta di Nonce e sulla Risposta di Nonce (*WP_131* (pagina 814)).

13.1.3 Endpoint di Gestione dell'Istanza di Wallet

Questi endpoint API RESTful sono forniti dal Fornitore di Wallet e consentono la gestione dell'Istanza di Wallet, inclusa la registrazione, il recupero dello stato, la revoca su richiesta (ad esempio, da parte dell'Utente) e l'eliminazione. Le seguenti sezioni descrivono le richieste di registrazione, recupero dello stato e revoca, insieme alle relative risposte, gestite da questo endpoint, che sono necessarie per le funzionalità di base dell'*Funzionalità dell'Istanza del Wallet* (pagina 303).

Richiesta di Registrazione dell'Istanza di Wallet

Per registrare un'Istanza di Wallet, la richiesta al Fornitore di Wallet DEVE utilizzare il metodo HTTP POST con `Content-Type` impostato su *application/json*. Il corpo della richiesta DEVE contenere i claim descritti in *Richiesta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 712) (*WP_131–134* (pagina 814)).

Risposta alla Registrazione dell'Istanza di Wallet

Se una Richiesta di Registrazione dell'Istanza di Wallet viene convalidata con successo, il Fornitore di Wallet fornisce una Risposta HTTP con codice di stato 204 (No Content). Per i dettagli vedere *Risposta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 713) (*WP_135–137* (pagina 814)).

Richiesta di Recupero dell'Istanza di Wallet

Per recuperare tutte le Istanze di Wallet associate a un Utente, una richiesta DEVE essere inviata utilizzando il metodo HTTP GET al Fornitore di Wallet (*WP_145* (pagina 814)).

Nota

Per recuperare una specifica Istanza di Wallet, la richiesta DEVE includere l'ID dell'Istanza di Wallet come parametro di percorso.

Risposta al Recupero dell'Istanza di Wallet

Se una Richiesta di Recupero dell'Istanza di Wallet viene elaborata con successo, il Fornitore di Wallet DEVE restituire una Risposta HTTP con un codice di stato 200 (OK). Il corpo della risposta DEVE essere in formato JSON e includere le informazioni rilevanti dell'Istanza di Wallet, come il suo ID univoco, lo stato e la data di emissione. Quando si recuperano tutte le Istanze di Wallet, la risposta DEVE restituire un array contenente i dettagli di tutte le istanze associate (*WP_146* (pagina 814)).

Se si verificano errori durante il processo di recupero, DEVE essere restituita una risposta di errore. Fare riferimento a *Gestione degli Errori per la Gestione dell'Istanza di Wallet* (pagina 574) per i dettagli sui codici di errore e le descrizioni.

Di seguito è riportato un esempio non normativo di una risposta di errore:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store

{
  "error": "forbidden",
  "error_description": "User is not authorized to retrieve Wallet Instances."
}
```

Richiesta di Revoca dell'Istanza di Wallet

Per revocare un'Istanza di Wallet attiva, una richiesta di revoca DEVE essere inviata utilizzando il metodo HTTP PATCH con Content-Type impostato su `application/json`. Il corpo della richiesta DEVE contenere un parametro `status` impostato su `REVOKED` (*WP_147* (pagina 814)).

Nota

Mentre PATCH è il metodo consigliato, la richiesta di revoca PUÒ anche essere inviata utilizzando il metodo POST, a seconda delle preferenze di implementazione.

Risposta alla Revoca dell'Istanza di Wallet

Se una Richiesta di Revoca dell'Istanza di Wallet viene elaborata con successo, il Fornitore di Wallet fornisce una Risposta HTTP con un codice di stato 204 (No Content) come specificato in *WP_148* (pagina 814).

Se si verificano errori durante la Revoca dell'Istanza di Wallet, DEVE essere restituita una risposta di errore. Fare riferimento a *Gestione degli Errori per la Gestione dell'Istanza di Wallet* (pagina 574) per i dettagli sui codici di errore e le descrizioni (*WP_035–039, WP_043–044* (pagina 781)).

Di seguito è riportato un esempio non normativo di una risposta di errore:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "bad_request",
  "error_description": "The request is missing status parameter."
}
```

Gestione degli Errori per la Gestione dell'Istanza di Wallet

Per garantire robustezza e sicurezza, il Fornitore di Wallet DEVE gestire gli errori in modo coerente in tutte le richieste di Gestione dell'Istanza di Wallet, incluse Registrazione, Recupero e Revoca (*WP_035–044* (pagina 781), e *WP_150–155* (pagina 814)).

In caso di errore, il Fornitore di Wallet DEVE restituire una risposta di errore come definito in **RFC 7231**, con ulteriori dettagli disponibili in **RFC 7807**. La risposta DEVE utilizzare il Content-Type impostato su `application/json` e DEVE includere i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Le seguenti sezioni classificano gli errori in **errori comuni**, che si applicano a tutte le richieste, ed **errori specifici della richiesta**, che sono rilevanti per operazioni particolari.

Risposte di Errore Comuni

I seguenti errori si applicano a tutte le operazioni di Gestione dell'Istanza di Wallet (Registrazione, Recupero e Revoca) e DEVONO essere supportati per la risposta di errore, se non diversamente specificato (*WP_035–039* (pagina 781)):

Codice di Stato HTTP	Codice di Errore	Descrizione
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri richiesti o include parametri non validi e sconosciuti.
`422 Unprocessable Content` [OPZIONALE]	`validation_error`	La richiesta non aderisce al formato richiesto.
`500 Internal Server Error`	`server_error`	Si è verificato un errore interno durante l'elaborazione della richiesta.
`503 Service Unavailable`	`temporarily_unavailable`	Il servizio non è disponibile. Si prega di riprovare più tardi.

Risposte di Errore Specifiche della Richiesta

Gli errori in *Risposta di Errore di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 713) DEVONO essere supportati per le risposte di errore relative alla **Registrazione dell'Istanza di Wallet**.

I seguenti errori DEVONO essere supportati per le risposte di errore relative al **Recupero dell'Istanza di Wallet** (*WP_041–042* (pagina 781)):

Codice di Stato HTTP	Codice di Errore	Descrizione
`403 Forbidden`	`forbidden`	L'Utente non ha il permesso di recuperare questa Istanza di Wallet.
`401 Unauthorized`	`unauthorized`	La richiesta manca di Credenziali di autenticazione valide.

I seguenti errori DEVONO essere supportati per le risposte di errore relative alla **Revoca dell'Istanza di Wallet** (*WP_043–044* (pagina 781)):

Codice di Stato HTTP	Codice di Errore	Descrizione
`403 Forbidden`	`invalid_request`	L'Utente non ha il permesso di revocare questa Istanza di Wallet.
`401 Unauthorized`	`unauthorized`	La richiesta non può essere autenticata o autorizzata.

13.1.4 Endpoint di Emissione della Wallet Instance Attestation

Questo è un endpoint API RESTful fornito dal Fornitore di Wallet che consente all'Istanza di Wallet di ottenere una Wallet Instance Attestation, inviando una Richiesta di Emissione della Wallet Instance Attestation.

Richiesta di Emissione della Wallet Instance Attestation

La richiesta di emissione della Wallet Instance Attestation utilizza il metodo HTTP POST con il `Content-Type` impostato su `application/json` (*WP_026* (pagina 781) e *WP_140–142* (pagina 814)).

L'intestazione `typ` del JWT della richiesta di emissione della Wallet Instance Attestation assume il valore `wia-request+jwt`.

Il body della richiesta di emissione della Wallet Instance Attestation contiene un parametro `assertion` il cui valore è un JWT firmato che include tutti i parametri di intestazione e le claim descritti di seguito.

Di seguito è riportato un esempio non normativo di una richiesta di emissione della Wallet Instance Attestation.

```
POST /wallet-instance-attestation HTTP/1.1
Host: application-provider.example.org
Content-Type: application/json

{
  "assertion": "eyJpc3MiOiJPbnNpYW5kcklqcDdJbU55ZGlJNklsQ..."
}
```

In particolare, il JWT della richiesta di emissione della Wallet Instance Attestation include i seguenti parametri di intestazione HTTP:

Parametro	Descrizione	Riferimento
alg	Identificatore dell'algoritmo di firma digitale, come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati in *Algoritmi Crittografici* (pagina 647) e non DEVE essere impostato su `none` né su alcun identificatore di algoritmo simmetrico (MAC).	[**RFC 7516#section-4.1.1**]
kid	thumbprint della JWK dell'Istanza del Wallet contenuta nella dichiarazione `cnf`.	[**RFC 7638#section_3**]
typ	Il tipo del JWT, che DEVE essere impostato su `wia-request+jwt`.	

Il JWT della richiesta include le seguenti claim nel body:

Claim	Descrizione	Riferimento
iss	Stringa contenente l'identificativo univoco della Wallet Instance.	[**RFC 9126**], [**RFC 7519**].
exp	UNIX timestamp che rappresenta il tempo di scadenza del JWT.	[**RFC 9126**], [**RFC 7519**].
iat	UNIX timestamp che rappresenta il tempo di emissione del JWT.	[**RFC 9126**], [**RFC 7519**].
nonce	Il `nonce` ottenuto dal Nonce Endpoint.	
hardware_signature	la firma di `client_data_hash` ottenuta utilizzando la Chiave Crittografica Hardware, codificata nel formato `base64url`.	
integrity_assertion	L'Integrity Assertion per la Wallet Instance Attestation ottenuta dalle **Device Integrity Service APIs** con il binding di `client_data_hash`.	
hardware_key_tag	Il valore del Tag della Chiave Crittografica Hardware.	
cnf	Oggetto JSON contenente la parte pubblica di una coppia di chiavi asimmetriche posseduta dall'istanza del Wallet.	**RFC 7800**.
platform	Stringa contenente il valore del sistema operativo del dispositivo.	
wallet_solution_id	Stringa contenente l'identificatore della Wallet Solution.	
wallet_solution_version	Stringa contenente la versione della Wallet Solution.	

Di seguito è riportato un esempio non normativo dell'intestazione e del payload del JWT della Wallet Instance Attestation Request.

```
{
  "alg": "ES256",
  "kid": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
```

```
  "typ": "wia-request+jwt"
}

{
  "iss": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
  "hardware_signature": "KoZIhvcNAQcCoIAwgAIB...",
  "integrity_assertion": "o2NmbXRvYXBwbGUtYXBwYXNzZXJ0aW9uLXBheWxvYWQtYXBw...",
  "hardware_key_tag": "QW12DylRTmF89iGkpydNDWW7m8bVpa2Fn9KBeXGYtfX",
  "cnf": {
    "jwk": {
      "crv": "P-256",
      "kty": "EC",
      "x": "8FJtI-yr3pjyRKGMnz4WmdnQD_uJSq4R95Nj98b44",
      "y": "MKZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg"
    }
  },
  "platform": "iOS",
  "wallet_solution_id": "Wallet-mobile",
  "wallet_solution_version": "1.1.0"
}
```


Risposta all'Emissione della Wallet Instance Attestation

Se la Richiesta di Emissione della Wallet Instance Attestation viene convalidata con successo, il Fornitore di Wallet restituisce una risposta HTTP con un codice di stato `200 OK` e Content-Type `application/json`. L'Oggetto JSON restituito DEVE possedere il parametro `wallet_instance-attestations` (vedi *Emissione della Wallet Instance Attestation* (pagina 305)). `wallet_insatnce_attestation` sono oggetti JSON che contengono rispettivamente la Wallet Instance Attestation. Entrambe le attestazioni sono firmate dal Fornitore di Wallet (*WP_027–029* (pagina 781) e *WP_143–144* (pagina 814)). La Wallet Instance Attestation JWT deve essere utilizzato per la fase di Emissione di un Attestato Elettronico, come OAuth Client Attestation, e sarà inviato al Fornitore di Attestati Elettronici come discusso in *Emissione di Attestati Elettronici* (pagina 443).

L'Oggetto JSON restituito nella risposta ha il seguente claim:

Parametro	Descrizione	Riferimento
wallet_instance-attestation	OBBLIGATORIO. Una stringa contenente una Wallet Instance Attestation.	Questa specifica.

I valori di `wallet_instance_attestation` sono stringha che contenente rispettivamento la Wallet Instance Attestation in formato JWT.

Se si verifica un errore durante il processo, viene restituita una risposta di errore. La risposta utilizza `application/json` come `Content-Type` e include i seguenti parametri:

- *error*: il codice di errore.
- *error_description*: testo in formato leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato (*WP_035* (pagina 781)).

Di seguito è riportato un esempio non normativo di una Wallet Instance Attestation Issuance Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The provided challenge is invalid, expired, or already used."
}
```

La seguente tabella elenca i codici di stato HTTP e i relativi codici di errore supportati per la risposta di errore, salvo diversa indicazione (*WP_036–039* (pagina 781) e *WP_150–155* (pagina 814)):

HTTP Status Code	Error Code	Descrizione
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri obbligatori (ad esempio parametri di intestazione, o Integrity Assertion), oppure include parametri non validi o sconosciuti.
`403 Forbidden`	`invalid_request`	L'istanza del Wallet è stata revocata.
`403 Forbidden`	`integrity_check_error`	Il dispositivo non soddisfa i requisiti minimi di sicurezza del Fornitore di Wallet.
`403 Forbidden`	`invalid_request`	La firma della Wallet Instance Attestation Request non è valida oppure non corrisponde alla chiave pubblica (JWK) associata.
`403 Forbidden`	`invalid_request`	La validazione della Integrity Assertion non è riuscita; la Integrity Assertion è stata manomessa o firmata in modo non corretto.
`403 Forbidden`	`invalid_request`	Il `nonce` fornito non è valido, è scaduto oppure è già stato utilizzato.
`403 Forbidden`	`invalid_request`	La Proof of Possession (`hardware_signature`) non è valida.
`403 Forbidden`	`invalid_request`	Il parametro `iss` non corrisponde all'identificatore URL previsto dal Fornitore di Wallet.
`404 Not Found`	`not_found`	L'istanza del Wallet non è stata trovata.
`422 Unprocessable Content` [OPZIONALE]	`validation_error`	La richiesta non rispetta il formato richiesto.

continues on next page

Tabella 13.7 – continua dalla pagina precedente

HTTP Status Code	Error Code	Descrizione
`500 Internal Server Error`	`server_error`	Si è verificato un errore interno del server durante l'elaborazione della richiesta.
`503 Service Unavailable`	`temporarily_unavailable`	Il servizio non è disponibile. Si prega di riprovare più tardi.

JWT della Wallet Instance Attestation

L'header JOSE del JWT della Wallet Instance Attestation contiene i seguenti parametri:

Header JOSE	Descrizione	Riferimento
alg	OBBLIGATORIO. Un identificatore di algoritmo di firma digitale come da registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` o qualsiasi identificatore di algoritmo simmetrico (MAC).	**RFC 7516# section-4.1.1**.
kid	OBBLIGATORIO. Identificatore univoco della chiave pubblica associata alla chiave privata che il Fornitore di Wallet ha utilizzato per firmare la Wallet Instance Attestation.	**RFC 7638# section_3**.
typ	OBBLIGATORIO. DEVE essere impostato su `oauth-client-attestation+jwt`	OPENID4VC-HAIP.
trust_chain	OPZIONALE. Sequenza di Entity Statement che compone la Catena di Fiducia relativa al Fornitore di Wallet.	OID-FED Sezione 4.3 *Trust Chain Header Parameter*.
x5c	OBBLIGATORIO. Contiene il certificato di chiave pubblica X.509 o la catena di certificati (**RFC 5280**) corrispondente alla chiave utilizzata per firmare digitalmente il JWT.	**RFC 7515** Sezione 4.1.8, SD-JWT-VC Sezione 3.5 e OPENID4VC-HAIP.

Il corpo del JWT della Wallet Instance Attestation contiene i seguenti claim:

Claim	Descrizione	Riferimento
iss	OBBLIGATORIO. Stringa contenente l'URL che identifica il Fornitore di Wallet.	**RFC 7519**.
sub	OBBLIGATORIO. JWK Thumbprint della chiave pubblica inclusa nel claim `cnf`.	**RFC 7519** e EUDI-TS 3.

continues on next page

Tabella 13.9 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
iat	OBBLIGATORIO. Timestamp UNIX con il tempo di emissione del JWT.	**RFC 9126** e **RFC 7519**.
exp	OBBLIGATORIO. Timestamp UNIX con il tempo di scadenza del JWT. La durata è specificata in *la durata della Wallet Instance Attestation* (pagina 290).	**RFC 9126** e **RFC 7519** e EUDI-TS 3.
nbf	OPZIONALE. Timestamp UNIX con data e orario prima del quale il JWT NON DEVE essere accettato.	**RFC 9126** e **RFC 7519**.
cnf	OBBLIGATORIO. Oggetto JSON, contenente la parte pubblica di una coppia di chiavi asimmetriche posseduta dall'Istanza di Wallet.	**RFC 7800**.
wallet_link	OBBLIGATORIO. Stringa contenente un URL per ottenere ulteriori informazioni sul Wallet e sul Fornitore di Wallet.	OpenID4VCI.
wallet_name	OBBLIGATORIO. Stringa contenente un nome leggibile dall'uomo del Wallet.	OpenID4VCI.

Di seguito è riportato un esempio non normativo dell'header e del payload della Wallet Instance Attestation JWT senza codifica e firma applicata:


```
{
    "typ": "oauth-client-attestation+jwt",
    "x5c": [
      "MIICrDCCAlKgAwIBAgIUCQ5zD8eGxAN6c5isgQk0jkp3qzwwCgYIKoZIzj0EAwIwgZIxCzAJBgNVBAYTAklUMQ4w
DAYDVQQIDAVMYXppbzENMAsGA1UEBwwEUm9tYTEWMBQGA1UECgwNUGFnb1BBIFMucC5BLjEkMCIGA1UEAwwbZm9vMTEuYmx
vYi5jb3JlLndpbmRvd3MubmV0MSYwJAYJKoZIhvcNAQkBFhdwYWdvcGFzcGFAcGVjLnBhZ29wYS5pdDAeFw0yNTA3MDMxNT
E2NTlaFw0yNjA3MDMxNTE2NTlaMIGSMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNVBAcMBFJvbWExFjAUB
gNVBAoMDVBhZ29QQSBTLnAuQS4xJDAiBgNVBAMMG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDEmMCQGCSqGSIb3DQEJ
ARYXcGFnb3Bhc3BhQHBlYy5wYWdvcGEuaXQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARtdnuBd5hAg3cPyz7/o8vBFyL
4sp45HikFogMZse7f9/iL4gn6YM/ehP1CQI0dFnT3c/RdxFyRukKWXscopwXso4GDMIGAMAkGA1UdEwQCMAAwCwYDVR0PBA
QDAgWgMCYGA1UdEQQfMB2CG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDAdBgNVHQ4EFgQUOd7M/0bo1+saPT/H9M+G9
6sKnYIwHwYDVR0jBBgwFoAUkKHxQLaudIm0WfwZeN5HMTG8SCMwCgYIKoZIzj0EAwIDSAAwRQIgSXKs9Gx2bXVG9nxap1I/
KqwHdh6SQDKI85J9n7FqrI8CIQCqltHTw6sci1R4RSOFySEvzEohQTbNTke+X2lD9iQd5A==",
```

```
      "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0OTMzWhcNMjcwNjA0MTI0OTMzWjCBkjELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMRYwFAYDVQQKDA1QYWdvUEEgUy5wLkEuMSQwIgYDVQQDDBtmb28xMS5ibG9iLmNvcmUud2luZG93
cy5uZXQxJjAkBgkqhkiG9w0BCQEWF3BhZ29wYXNwYUBwZWMucGFnb3BhLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA
EgXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l/iX7p1ysisVV2vOhj1xEm/3M24U5HcywPPpNIOEHTESjqOCAW4wgg
FqMB0GA1UdDgQWBBSQofFAtq50ibRZ/Bl43kcxMbxIIzCB5QYDVR0jBIHdMIHagBTzUINxjECpXem0zyIpsnTqv3DEGaGBu
aSBtjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQ
b2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYDVQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHp
zLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5pcHpzLml0ggYBlzr5eUMwEgYDVR0TAQH/BAgwBgEB/wIBAD
AOBgNVHQ8BAf8EBAMCAAcwPQYDVR0fBDYwNDAyoDCgLoYsaHR0cHM6Ly9wcmUudGEud2FsbGV0LmlwenMuaXQvcGtpL3RhL
XN1Yi5jcmwwCgYIKoZIzj0EAwIDRwAwRAIgBSyH++oPDP8ipRxHcJ8u2HkN7wL44XpJFxSIBV9RFHMCIBixpcT43NSsk0Ev
hJb4g68c1d9OrBnXOXVuyCtWWE6a",
      "MIIDQzCCAuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0NTE3WhcNMzAwNjAzMTI0NTE3WjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQb2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYD
VQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHpzLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5
pcHpzLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEaE0xyhd3e9LDT7uwHOclL5H3389gwiCwFhI3KOvidn0glBIHYx
qH+4Z9VTMYWG5L8cwC9AaJUCNGu2dp5ZiiTKOB5TCB4jAdBgNVHQ4EFgQU81CDcYxAqV3ptM8iKbJ06r9wxBkwHwYDVR0jB
BgwFoAU81CDcYxAqV3ptM8iKbJ06r9wxBkwDwYDVR0TAQH/BAUwAwEB/zBEBggrBgEFBQcBAQQ4MDYwNAYIKwYBBQUHMAKG
KGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jZXIwDgYDVR0PAQH/BAQDAgEGMDkGA1UdHwQyMDAwLqA
soCqGKGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jcmwwCgYIKoZIzj0EAwIDSQAwRgIhAOsQYzR+eG
f4je63VGHqkpmkBbfyOre+mfIdHHowWWR/AiEA58xBNb5UW5uMB+tQur8fq24RD5MmRHLYS6bDgIYmluw="
    ],
    "alg": "ES256",
    "kid": "8UkfrvttLkpAQOOp"
  }

{
    "cnf": {
      "jwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "8g9qKk3tRCBHsu57pNy3oMPc-jeJW8zU1LvfG9H5__M",
        "y": "KD0MNCPtJqUd1RGiTdAxEL6OCw5trfzqSrU6BHKOy2g",
        "kid": "dd762e6f8b613193",
        "alg": "ES256"
      }
    },
    "iss": "https://wallet-provider.example.org",
    "sub": "dd762e6f8b613193",
    "wallet_name": "ExampleOrg",
    "wallet_link": "https://example.org/wallets/ExampleOrg/info",
    "iat": 1772444318,
```

```
  "exp": 1772447918
}
```

Nota

Come meccanismo di revoca per la WIA, è preferita l'opzione di riutilizzo per emittente descritta nella Sezione 2.5.1 di EUDI-TS 3.

13.1.5 Endpoint di Emissione della Key Attestation

Questo è un endpoint API RESTful fornito dal Wallet Provider che consente alla Wallet Instance di ottenere una Key Attestation, inviando una Richiesta di Emissione della Key Attestation.

Richiesta di Emissione della Key Attestation

La richiesta di emissione della Key Attestation utilizza il metodo HTTP POST con `Content-Type` impostato a `application/json`. (*WP_026* (pagina 781) e *WP_140–142* (pagina 814)).

L'header `typ` del JWT della richiesta di emissione Key Attestation Issuance assume il valore `wua-request+jwt`.

Il body della richiesta di emissione Key Attestation contiene un parametro `assertion` il cui valore è un JWT firmato che include tutti i parametri di header e i claim del body descritti di seguito.

Di seguito è riportato un esempio non normativo di una Key Attestation Request.

```
POST /key-attestation HTTP/1.1
Host: application-provider.example.org
Content-Type: application/json

{
  "assertion": "eyJpc3MiOiJPbnNpYW5kcklqcDdJbU55ZGlJNklsQ..."
}
```

In particolare, il JWT della Key Attestation Issuance include i seguenti parametri di header HTTP:

Parameter	Description	Reference
alg	Identificatore di algoritmo di firma digitale, come definito dal registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati in *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato a `none` né a un identificatore di algoritmo simmetrico (MAC).	[**RFC 7516#section-4.1.1**]
kid	Thumbprint della JWK della Wallet Instance contenuta nel claim `cnf`.	[**RFC 7638#section_3**]
typ	Il tipo del JWT; DEVE essere impostato a `wua-request+jwt`.	

Il JWT della Key Attestation Request include i seguenti claim nel body:

Claim	Description	Reference
iss	Stringa contenente l'identificativo univoco della Wallet Instance.	[**RFC 9126**], [**RFC 7519**].
exp	Timestamp UNIX che rappresenta il tempo di scadenza del JWT.	[**RFC 9126**], [**RFC 7519**].
iat	Timestamp UNIX che rappresenta il tempo di emissione del JWT.	[**RFC 9126**], [**RFC 7519**].
nonce	Il `nonce` ottenuto dal Nonce Endpoint.	
keys_to_attest	Array JSON di stringhe JWT, ciascuna delle quali rappresenta una `Key_Attestation_Requests`.	
hardware_signature	La firma di `client_data_hash` ottenuta utilizzando la Cryptographic Hardware Key, codificata nel formato `base64url`.	
integrity_assertion	L'Integrity Assertion per la Wallet Instance Attestation ottenuta dalle **Device Integrity Service APIs** con holder binding di `client_data_hash`.	
hardware_key_tag	Il valore del Cryptographic Hardware Key Tag.	
cnf	Oggetto JSON contenente la parte pubblica della prima coppia di chiavi asimmetriche (primo elemento di `keys_to_attest`) posseduta dalla Wallet Instance.	**RFC 7800**.
platform	Stringa contenente il valore del sistema operativo del dispositivo.	
wallet_solution_id	Stringa contenente l'identificatore della Wallet Solution.	
wallet_solution_version	Stringa contenente la versione della Wallet Solution.	

Di seguito è riportato un esempio non normativo dell'header e del payload JWT di una Key Attestation Request.

```
{
  "alg": "ES256",
  "kid": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "typ": "wua-request+jwt"
}
```

```
{
  "iss": "OnsiandrIjp7ImNydiI6IlAtMjU2Iiwia3R5IjoiRUMiL",
  "nonce": "f3b29a81-45c7-4d12-b8b5-e1f6c9327aef",
  "hardware_signature": "KoZIhvcNAQcCoIAwgAIB...",
  "integrity_assertion": "o2NmbXRvYXBwbGUtYXBwYXNzZXJ0aW9uLXBheWxvYWQtYXBw...",
  "hardware_key_tag": "QW12DylRTmF89iGkpydNDWW7m8bVpa2Fn9KBeXGYtfX",
  "cnf": {
    "jwk": {
      "crv": "P-256",
      "kty": "EC",
      "x": "8FJtI-yr3pjyRKGMnz4WmdnQD_uJSq4R95Nj98b44",
      "y": "MKZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg"
    }
  },
  "keys_to_attest": [
    "eyJ0eXAiOiJrZXktYXR0ZXN0YXRpb24tcmVxdWVzdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6Ik9LSEhrVk5Pck
thUFZKdWZsREt3MVNRSEZOWTVpeTlPaXdBdHBBMGNvSUEifQ.eyJ3c2NkX2tleV9hdHRlc3RhdGlvbiI6eyJzdG9yYWdlX3
R5cGUiOiJMT0NBTF9OQVRJVkUifSwiY25mIjp7Imp3ayI6eyJrdHkiOiJFQyIsIngiOiJ4QUg5U05mYXE5SjVkbWt6WFlRT
GVrNVlmcFBjOGlfUHBNUlQzMTVoak1rIiwieSI6IlBFMlhMY3BXNmVYSDRGbFlHTlA5Qmh3UVFkRWlaRTF0QWRULUVpaEFD
QzgiLCJjcnYiOiJQLTI1NiIsImtpZCI6Ik9LSEhrVk5PckthUFZKdWZsREt3MVNRSEZOWTVpeTlPaXdBdHBBMGNvSUEifX0
sImlhdCI6MTc3MzA1Mzg2MSwiZXhwIjoxNzczMDU3NDYxfQ.Rn3D0GwYYZJaupzJ6617V0xav_HH6bGnttoGrD4lwY8ICPH
9NiEbTF9ZBYD3aHh20Z9GCjQ8Fhit5Fbps8v9Aw",
    "eyJ0eXAiOiJrZXktYXR0ZXN0YXRpb24tcmVxdWVzdCtqd3QiLCJhbGciOiJFUzI1NiIsImtpZCI6IkViUUJSQ2dLNW
JrVzlZNU1idGEwZlpzMVdhVTBLZVpiek9iTXVvY2NLb28ifQ.eyJ3c2NkX2tleV9hdHRlc3RhdGlvbiI6eyJzdG9yYWdlX3
R5cGUiOiJMT0NBTF9OQVRJVkUifSwiY25mIjp7Imp3ayI6eyJrdHkiOiJFQyIsIngiOiJEVVFWTGhLMUtRUmQtZ3g3UU5jY
VNhWENnOXg0S3R6QmstNWIxWTNkeWU0IiwieSI6IkZxVjk0TWVrVm5fQ05mNTIxdm1vLVFIcWZObk12eGdIR3NFeDlCTlc4
aFEiLCJjcnYiOiJQLTI1NiIsImtpZCI6IkViUUJSQ2dLNWJrVzlZNU1idGEwZlpzMVdhVTBLZVpiek9iTXVvY2NLb28ifX0
sImlhdCI6MTc3MzA1Mzg2MSwiZXhwIjoxNzczMDU3NDYxfQ.wIYOmX8-dmuRnuaCVg1kFoTHhsvv01vbapQ8-3er-HIiAF8
19Kt3Uy0PUN_WgxP7eWMGwhkn_9tQnnhdgXLYyw"
  ],
  "platform": "iOS",
  "wallet_solution_id": "Wallet-mobile",
  "wallet_solution_version": "1.1.0"
}
```

Risposta all'Emissione della Key Attestation

Se la Key Attestation Issuance Request viene validata con successo, il Wallet Provider restituisce una risposta HTTP con codice di stato `200 OK` e `Content-Type application/json`. L'oggetto JSON restituito include `key_attestation` (si veda *Emissione della Key Attestation* (pagina 308)). `key_attestation` è firmata dal Wallet Provider (*WP_027–029* (pagina 781) e *WP_143–144* (pagina 814)). La Key Attestation JWT deve essere utilizzata nella fase di Emissione di un Attestato Elettronico, come intestazione JOSE `key_attestation` nel JWT di tipo `proof`, e verrà inviata al Fornitore di Attestati Elettronici come discusso in *Emissione di Attestati Elettronici* (pagina 443).

L'oggetto JSON restituito nella risposta contiene il seguente parametro:

Parameter	Description	Reference
key_attestation	OBBLIGATORIO. Una stringa che rappresenta la Key Attestation rilasciata.	Questa specifica.

Il valore del parametro `key_attestation` è una stringa che rappresenta la Key Attestation in formato JWT.

Se durante il processo si verificano errori, viene restituita una risposta di errore come definito nella sezione precedente.

La tabella seguente elenca i codici di stato HTTP e i relativi codici di errore per i casi che differiscono da quanto già riportato:

HTTP Status Code	Error Code	Description
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri obbligatori (ad esempio parametri di header, Integrity Assertion o `keys_to_attest`), oppure include parametri non validi o sconosciuti.
`403 Forbidden`	`invalid_request`	La validazione dell'Integrity Assertion o della Key Attestation (`keys_to_attest`) non è riuscita; l'Integrity Assertion o la Key Attestation (`keys_to_attest`) è stata manomessa oppure firmata in modo non corretto.
`403 Forbidden`	`invalid_request`	La firma della key Attestation Request non è valida oppure non corrisponde alla chiave pubblica associata (JWK).

Key Attestation JWT

L'intestazione JOSE del Key Attestation JWT contiene i seguenti parametri:

JOSE header	Descrizione	Riferimento
alg	OBBLIGATORIO. Un identificatore di algoritmo di firma digitale come da registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati in *Algoritmi Crittografici* (pagina 647) e NON DEVE essere impostato su `none` o qualsiasi identificatore di algoritmo simmetrico (MAC).	**RFC 7516# section-4.1.1**.
kid	OBBLIGATORIO. Identificatore univoco della chiave pubblica associata alla chiave privata che il Fornitore di Wallet ha utilizzato per firmare la Key Attestation.	**RFC 7638# section_3**.
typ	OBBLIGATORIO. DEVE essere impostato su `key-attestation+jwt`	OPENID4VC-HAIP.
trust_chain	OPZIONALE. Sequenza di Entity Statement che compone la Catena di Fiducia relativa al Fornitore di Wallet.	OID-FED Sezione 4.3 *Trust Chain Header Parameter*.
x5c	OBBLIGATORIO. Contiene il certificato di chiave pubblica X.509 o la catena di certificati (**RFC 5280**) corrispondente alla chiave utilizzata per firmare digitalmente il JWT.	**RFC 7515** Sezione 4.1.8 e SD-JWT-VC Sezione 3.5.

Il corpo del Key Attestation JWT contiene le seguenti dichiarazioni (claims):

Claim	Descrizione	Riferimento
iss	OBBLIGATORIO. Stringa contenente l'URL che identifica il Fornitore di Wallet.	**RFC 7519**.
exp	OBBLIGATORIO. Timestamp UNIX con il tempo di scadenza del JWT.	**RFC 9126** e **RFC 7519**.
iat	OBBLIGATORIO. Timestamp UNIX con il tempo di emissione del JWT.	**RFC 9126** e **RFC 7519**.
attested_keys	OBBLIGATORIO. Un array non vuoto di chiavi attestate provenienti dallo stesso componente di archiviazione delle chiavi, utilizzando la sintassi JWK, contenente la parte pubblica di una coppia di chiavi asimmetriche posseduta dall'istanza del Wallet.	**RFC 7517**.

continues on next page

Tabella 13.15 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
key_storage	OBBLIGATORIO. Un array non vuoto di stringhe sensibili alle maiuscole che dichiarano la resistenza al potenziale di attacco del componente di archiviazione delle chiavi e delle chiavi attestate nel parametro `attested_keys`. I seguenti valori sono definiti come possibili valori per questa dichiarazione claim: • `iso_18045_high`: DEVE essere utilizzato quando l'archiviazione delle chiavi è resistente ad attacchi con potenziale di attacco `High`. • `iso_18045_moderate`: DEVE essere utilizzato quando l'archiviazione delle chiavi è resistente ad attacchi con potenziale di attacco `Moderate`. • `iso_18045_enhanced-basic`: DEVE essere utilizzato quando l'archiviazione delle chiavi è resistente ad attacchi con potenziale di attacco `Enhanced-Basic`. • `iso_18045_basic`: DEVE essere utilizzato quando l'archiviazione delle chiavi è resistente ad attacchi con potenziale di attacco `Basic`.	OpenID4VCI.
user_ authentication	OBBLIGATORIO. Un array non vuoto di stringhe sensibili alle maiuscole che dichiarano la resistenza al potenziale di attacco dei metodi di autenticazione utente autorizzati ad accedere alle chiavi private dal parametro attested_keys. I seguenti valori sono definiti come possibili valori per questa dichiarazione claim: • `iso_18045_high`: DEVE essere utilizzato quando l'autenticazione dell'utente è resistente ad attacchi con potenziale di attacco `High`. • `iso_18045_moderate`: DEVE essere utilizzato quando l'autenticazione dell'utente è resistente ad attacchi con potenziale di attacco `Moderate`. • `iso_18045_enhanced-basic`: DEVE essere utilizzato quando l'autenticazione dell'utente è resistente ad attacchi con potenziale di attacco `Enhanced-Basic`. • `iso_18045_basic`: DEVE essere utilizzato quando l'autenticazione dell'utente è resistente ad attacchi con potenziale di attacco `Basic`.	OpenID4VCI.

continues on next page

Tabella 13.15 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
status	OBBLIGATORIO. Meccanismo di stato per la Key Attestation. • **status_list**: OBBLIGATORIO. un riferimento alla lista di stato, come specificato nell'Appendice D di OpenID4VCI. Il valore rappresenta lo stato di revoca del WSCD o del Keystore.	EUDI-TS 3.
certification	OPZIONALE. Una stringa che contiene un URL che rimanda alla certificazione del componente di archiviazione delle chiavi.	OpenID4VCI.

Il Fornitore di Wallet DEVE impostare `key_storage` e `user_authentication` su valori ISO 18045 che corrispondono alla resistenza al potenziale di attacco del Keystore o WSCD effettivo e del metodo di autenticazione dell'Utente. Il Fornitore di Wallet DEVE documentare il mapping usato per TEE, StrongBox e Secure Enclave. Un Keystore Interno Locale NON DEVE essere attestato con `iso_18045_high`. Un WSCD certificato come resistente ad attaccanti con potenziale di attacco elevato (ad esempio Common Criteria EAL4+ AVA_VAN.5) PUÒ essere attestato con `iso_18045_high` (*WP_022a* (pagina 781)).

Di seguito è riportato un esempio non normativo dell'intestazione e del payload del Key Attestation JWT, senza codifica né firma applicata:


```
{
    "typ": "key-attestation+jwt",
    "x5c": [
      "MIICrDCCAlKgAwIBAgIUCQ5zD8eGxAN6c5isgQk0jkp3qzwwCgYIKoZIzj0EAwIwgZIxCzAJBgNVBAYTAklUMQ4w
DAYDVQQIDAVMYXppbzENMAsGA1UEBwwEUm9tYTEWMBQGA1UECgwNUGFnb1BBIFMucC5BLjEkMCIGA1UEAwwbZm9vMTEuYmx
vYi5jb3JlLndpbmRvd3MubmV0MSYwJAYJKoZIhvcNAQkBFhdwYWdvcGFzcGFAcGVjLnBhZ29wYS5pdDAeFw0yNTA3MDMxNT
E2NTlaFw0yNjA3MDMxNTE2NTlaMIGSMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8xDTALBgNVBAcMBFJvbWExFjAUB
gNVBAoMDVBhZ29QQSBTLnAuQS4xJDAiBgNVBAMMG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDEmMCQGCSqGSIb3DQEJ
ARYXcGFnb3Bhc3BhQHBlYy5wYWdvcGEuaXQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARtdnuBd5hAg3cPyz7/o8vBFyL
4sp45HikFogMZse7f9/iL4gn6YM/ehP1CQI0dFnT3c/RdxFyRukKWXscopwXso4GDMIGAMAkGA1UdEwQCMAAwCwYDVR0PBA
QDAgWgMCYGA1UdEQQfMB2CG2ZvbzExLmJsb2IuY29yZS53aW5kb3dzLm5ldDAdBgNVHQ4EFgQUOd7M/0bo1+saPT/H9M+G9
6sKnYIwHwYDVR0jBBgwFoAUkKHxQLaudIm0WfwZeN5HMTG8SCMwCgYIKoZIzj0EAwIDSAAwRQIgSXKs9Gx2bXVG9nxap1I/
KqwHdh6SQDKI85J9n7FqrI8CIQCqltHTw6sci1R4RSOFySEvzEohQTbNTke+X2lD9iQd5A==",
```

```
      "MIIDqjCCA1GgAwIBAgIGAZc6/V9qMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0OTMzWhcNMjcwNjA0MTI0OTMzWjCBkjELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMRYwFAYDVQQKDA1QYWdvUEEgUy5wLkEuMSQwIgYDVQQDDBtmb28xMS5ibG9iLmNvcmUud2luZG93
cy5uZXQxJjAkBgkqhkiG9w0BCQEWF3BhZ29wYXNwYUBwZWMucGFnb3BhLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA
EgXY4FApFJCj91Gpb1K9GEIouTq2X3L0K64Iq0ob4l/iX7p1ysisVV2vOhj1xEm/3M24U5HcywPPpNIOEHTESjqOCAW4wgg
FqMB0GA1UdDgQWBBSQofFAtq50ibRZ/Bl43kcxMbxIIzCB5QYDVR0jBIHdMIHagBTzUINxjECpXem0zyIpsnTqv3DEGaGBu
aSBtjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvMQ0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQ
b2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYDVQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHp
zLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5pcHpzLml0ggYBlzr5eUMwEgYDVR0TAQH/BAgwBgEB/wIBAD
AOBgNVHQ8BAf8EBAMCAAcwPQYDVR0fBDYwNDAyoDCgLoYsaHR0cHM6Ly9wcmUudGEud2FsbGV0LmlwenMuaXQvcGtpL3RhL
XN1Yi5jcmwwCgYIKoZIzj0EAwIDRwAwRAIgBSyH++oPDP8ipRxHcJ8u2HkN7wL44XpJFxSIBV9RFHMCIBixpcT43NSsk0Ev
hJb4g68c1d9OrBnXOXVuyCtWWE6a",
      "MIIDQzCCAuigAwIBAgIGAZc6+XlDMAoGCCqGSM49BAMCMIGzMQswCQYDVQQGEwJJVDEOMAwGA1UECAwFTGF6aW8x
DTALBgNVBAcMBFJvbWExMTAvBgNVBAoMKElzdGl0dXRvIFBvbGlncmFmaWNvIGUgWmVjY2EgZGVsbG8gU3RhdG8xCzAJBgN
VBAsMAklUMR4wHAYDVQQDDBVwcmUudGEud2FsbGV0LmlwenMuaXQxJTAjBgkqhkiG9w0BCQEWFnByb3RvY29sbG9AcGVjLm
lwenMuaXQwHhcNMjUwNjA0MTI0NTE3WhcNMzAwNjAzMTI0NTE3WjCBszELMAkGA1UEBhMCSVQxDjAMBgNVBAgMBUxhemlvM
Q0wCwYDVQQHDARSb21hMTEwLwYDVQQKDChJc3RpdHV0byBQb2xpZ3JhZmljbyBlIFplY2NhIGRlbGxvIFN0YXRvMQswCQYD
VQQLDAJJVDEeMBwGA1UEAwwVcHJlLnRhLndhbGxldC5pcHpzLml0MSUwIwYJKoZIhvcNAQkBFhZwcm90b2NvbGxvQHBlYy5
pcHpzLml0MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEaE0xyhd3e9LDT7uwHOclL5H3389gwiCwFhI3KOvidn0glBIHYx
qH+4Z9VTMYWG5L8cwC9AaJUCNGu2dp5ZiiTKOB5TCB4jAdBgNVHQ4EFgQU81CDcYxAqV3ptM8iKbJ06r9wxBkwHwYDVR0jB
BgwFoAU81CDcYxAqV3ptM8iKbJ06r9wxBkwDwYDVR0TAQH/BAUwAwEB/zBEBggrBgEFBQcBAQQ4MDYwNAYIKwYBBQUHMAKG
KGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jZXIwDgYDVR0PAQH/BAQDAgEGMDkGA1UdHwQyMDAwLqA
soCqGKGh0dHBzOi8vcHJlLnRhLndhbGxldC5pcHpzLml0L3BraS90YS5jcmwwCgYIKoZIzj0EAwIDSQAwRgIhAOsQYzR+eG
f4je63VGHqkpmkBbfyOre+mfIdHHowWWR/AiEA58xBNb5UW5uMB+tQur8fq24RD5MmRHLYS6bDgIYmluw="
    ],
    "alg": "ES256",
    "kid": "8UkfrvttLkpAQOOp"
  }

{
    "attested_keys": [
        {
            "crv": "P-256",
            "kty": "EC",
            "x": "4HNptI-xr2pjyRJKGMnz4WmdnQD_uJSq4R95Nj98b44",
            "y": "LIZnSB39vFJhYgS3k7jXE4r3-CoGFQwZtPBIRqpNlrg",
            "kid": "vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c"
        }
    ],
    "iss": "https://wallet-provider.example.org",
    "key_storage": [ "iso_18045_moderate" ],
    "status": {
        "status_list": {
            "idx": 412,
            "uri": "https://revocation_url/statuslists/1"
```

```
        }
    },
    "user_authentication": [ "iso_18045_moderate" ],
    "iat": 1687281195,
    "exp": 1687288395
}
```

Nota

Poiché lo schema di certificazione non è ancora stato definito, il contenuto esatto di `certification` non è definito e sarà specificato in un futuro aggiornamento.

Nota

Come meccanismo di revoca per la KA, è preferito l'indice condiviso per tipo descritto nella Sezione 2.5.2 di EUDI-TS 3.

13.1.6 Catalogo e-Service PDND del Fornitore di Wallet

La morte dell'Utente porta alla revoca delle Istanze di Wallet dell'Utente e all'eliminazione dell'account dell'Utente presso il Fornitore di Wallet. Per questo motivo, il Fornitore di Wallet fornisce il seguente e-service tramite PDND. Un Provider di PID/IT-Wallet ID che è stato notificato dalla Fonte Autentica del PID/IT-Wallet ID della morte dell'Utente DEVE inviare una notifica ai Fornitori di Wallet utilizzando questo endpoint. Le chiamate al Catalogo e-Service PDND del Fornitore di Wallet, inclusa Notifica Morte Utente, DEVONO essere autenticate e autorizzate. Le richieste non autenticate DEVONO essere rifiutate. Una Notifica Morte Utente andata a buon fine DEVE portare alla revoca dell'Istanza del Wallet come per le altre revoche avviate dal Provider di PID/IT-Wallet ID (*WP_161* (pagina 781)).

Nota

Una Specifica OpenAPI completa è disponibile *Specifica OpenAPI del Fornitore di Wallet PDND* (pagina 879).

Notifica Morte Utente

Descrizione	Questo servizio viene utilizzato per notificare al Fornitore di Wallet la necessità di revocare l'Istanza di Wallet ed eliminare l'account dell'Utente a causa della morte dell'Utente.
Fornitore	Fornitore di Wallet
Consumatore	Fornitore di Attestati Elettronici di Dati di Identificazione Personale

13.2 Endpoint del Credential Issuer

13.2.1 Endpoint Metadata

I Credential Issuer DEVONO fornire una Entity Configuration attraverso l'endpoint `/.well-known/openid-federation`, secondo la Sezione *Entity Configuration* (pagina 126). I dettagli tecnici sono forniti nella Sezione credential-issuer-entity-configuration:Entity Configuration del Fornitore di Attestati Elettronici.

Alternativamente i metadati del Credential Issuer possono essere recuperati utilizzando l'identificativo del Credential Issuer. Il documento di metadata DEVE essere reso disponibile in formato JSON e JWT presso l'endpoint `/.well-known/openid-credential-issuer` come definito nella Sezione 12.2.2 di OpenID4VCI.

L'header `Accept-Language` nella richiesta HTTP GET può essere utilizzato per indicare la/le lingua/e preferite. In tal caso, il Credential Issuer può inviare un sottoinsieme dei metadati contenente dati di visualizzazione internazionalizzati per una o tutte le lingue richieste, e può indicare le lingue restituite utilizzando l'header HTTP `Content-Language`.

Di seguito è riportato un esempio non normativo.

```
GET /.well-known/openid-credential-issuer HTTP/1.1
Host: issuer.example.com
Accept: application/json
Accept-Language: it-IT, it;q=0.9

GET /.well-known/openid-credential-issuer HTTP/1.1
Host: issuer.example.com
Accept: application/jwt
Accept-Language: it-IT, it;q=0.9
```

Il Credential Issuer DEVE rispondere con il Status Code HTTP 200 e restituire i metadati del Credential Issuer contenenti:

- i parametri definiti nella sezione credential-issuer-metadata:Metadata per openid_credential_issuer all'interno di un documento JSON non firmato utilizzando il media type *application/json*.

  oppure

- i parametri di header e payload definiti nella sezione 12.2.3 di OpenID4VCI oltre a quelli definiti nella sezione credential-issuer-metadata:Metadata per openid_credential_issuer all'interno di un JSON Web Token firmato utilizzando il media type *application/jwt*.

Di seguito è riportato un esempio non normativo dei metadati del Credential Issuer firmati:

ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgInR5cCI6ICJvcGVuaWR2Y2ktaXNzdWVyLW1ldGFkYXRhK2p3dCIsDQogICJraW
QiOiAiZEI2N2dMN2NrM1RGaUlBZjdONl83U0h2cWswTURZTUVRY29HR2xrVUFBdyIgDQp9.ew0KCSJpYXQiOiAxNzE4MjA
3MjE3LA0KCSJleHAiOiAxNzQ5NzQzMjE2LA0KCSJpc3MiOiAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmciL
A0KCSJzdWIiOiAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmciLA0KCSJjcmVkZW50aWFsX2lzc3VlciI6ICJ
odHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZyIsDQoJImNyZWRlbnRpYWxfZW5kcG9pbnQiOiAiaHR0cHM6Ly9lY
WEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvY3JlZGVudGlhbCIsDQoJIm5vbmNlX2VuZHBvaW50IjogImh0dHBzOi8vZWFhLXB
yb3ZpZGVyLmV4YW1wbGUub3JnL25vbmNlLWVuZHBvaW50IiwNCgkiZGVmZXJyZWRfY3JlZGVudGlhbF9lbmRwb2ludCI6I
CJodHRwczovL2VhYS1wcm92aWRlci5leGFtcGxlLm9yZy9kZWZlcnJlZC1jcmVkZW50aWFsIiwNCgkibm90aWZpY2F0aW9
uX2VuZHBvaW50IjogImh0dHBzOi8vZWFhLXByb3ZpZGVyLmV4YW1wbGUub3JnL25vdGlmaWNhdGlvbiIsDQoJImRpc3BsY
XkiOiBbDQoJCXsNCgkJCSJuYW1lIjogIkVBQSBQcm92aWRlciIsDQoJCQkibG9jYWxlIjogIml0LUlUIiwNCgkJCSJsb2d
vIjogew0KCQkJCSJ1cmki0iAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvcHVibGljL2NvbXBhY
3QtbG9nby1VUy5zdmciLA0KCQkJCSJ1cmkjaW50ZWdyaXR5IjogIjSbHR5Z283YktCRTRFNDk4dGIvbExuOURJa2ltQmhyenJzVUV
SeVVRam89IiwNCgkJCQkiYWx0X3RleHQiOiAiVGVzdG8gYWx0ZXJuYXRpdm8gYWxsJ2ltbWFnaW5lIGRlbCBsb2dvIg0KC
QkJfQ0KDQoJCX0sDQoJCXsNCgkJCSJuYW1lIjogIkVBQSBQcm92aWRlciIsDQoJCQkibG9jYWxlIjogImVuLVVTIiwNCgk
JCSJsb2dvIjogew0KCQkJCSJ1cmki0iAiaHR0cHM6Ly9lYWEtcHJvdmlkZXIuZXhhbXBsZS5vcmcvcHVibGljL2NvbXBhY
3QtbG9nby1VUy5zdmciLA0KCQkJCSJ1cmkjaW50ZWdyaXR5IjogIm1CUUVpNDJhajBWWW5GbmdwT2picUpHa3VDbkI1TG1
iWlJuV3BPK0FIcVVCVlF4T2U5a01JUE96K3BTalJBckUiLA0KCQkJCSJhbHRfdGV4dCI6ICJBbHRlcm5hdGl2ZSB0ZXh0I
GZvciB0aGUgbG9nbyBpbWFnZSINCgkJCX0NCgkJfQ0KCV0sDQoJImNyZWRlbnRpYWxfY29uZmlndXJhdGlvbnMiOiBbDQoJ
ydGVkIjogew0KCQkiZGNfc2Rfand0X0V1cm9wZWFuRGlzYWJpbGl0eUNhcmQiOiB7DQoJCQkiZm9ybWF0IjogImRjK3NkL
Wp3dCIsDQoJCQkic2NvcGUiOiAiRXVyb3BlYW5EaXNhYmlsaXR5Q2FyZCIsDQoJCQkiY3J5cHRvZ3JhcGhpY19iaW5kaW5
nX21ldGhvZHNfc3VwcG9ydGVkIjogWw0KCQkJqd2siDQoJCQldLA0KCQkJImNyZWRlbnRpYWxfc2lnbmluZ19hbGdfdfd
mFsdWVzX3N1cHBvcnRlZCI6IFsNCgkJCSJFUzM4NCIsDQoJCQkJIkVTNEYig0KCQkJXSwNCgk
JCSJwcm9vZl90eXBlc19zdXBwb3J0ZWQiOiB7DQoJCQkJImp3dCI6IHsNCgkJCQkJInByb29mX3NpZ25pbmdfYlgiOiBb
zOi8vdHJ1c3QtcmVnaXN0cnkuXQtd2FsbGV0LmV4YW1wbGUuaXQvbG9nb3MvRXVyb3BlYW5EaXNhYmlsaXR5Q2FyZC9sb
2dvLUlULnN2ZyIsDQoJCQkJCQkJInVyaSNpbnRlZ3JpdHkiOiAiM0dHWkpIN2lna3llcHJYSTlQbG0zZFFzV01ZT2V1ejB
NYUVLMzRQQzdzT3RMZmdlb0s5c2NYT3ZNZWlBQ0Y4VGsyV09UMzZOc1lDUGR1b0x5T0QxU2c9PSIsDQoJCQkJImFsd
F90ZXh0IjogIlRlc3RvIGFsdGVybmF0aXZvIGFsbCdpbW1hZ2luZSBkZWwgbG9nbyINCgkJCQkJCQkiYmF
ja2dyb3VuZF9jb2xvciI6ICIjMTIxMDdjIiwNCgkJCQkJCQkiSiYWNrZ3JvdW5kX2ltYWdlIjogew0KDQoJCQkJCQkiY
SI6ICJodHRwczovL3RydXN0LXJlZ2lzdHJ5Lml0LXdhbGxldC5leGFtcGxlLml0L2ltYWdlcy9FdXJvcGVhbkRpc2FiaWx
pdHlDYXJkL2ltYWdlLUlULnN2ZyIsDQoJCQkJCQkJInVyaSNpbnRlZ3JpdHkiOiAiM0dHWkpIN2lna3llcHJYSTlQbG0zZ
FFzV01ZT2V1ejBNYUVLMzRQQzdzT3RMZmdlb0s5c2NYT3ZNZWlBQ0Y4VGsyV09UMzZOc1lDUGR1b0x5T0QxU2c9PSINCg0
KCQkJCQkJCQkJfSwNCgkJCQkJCQkJCSJ3YXRlcm1hcmtfaW1hZ2UiOiB7DQoNCgkJCQkJCQkJCQkidXJpIjogImh0dHBzOi8vdHJ1c3Qtc
mVnaXN0cnkuXQtd2FsbGV0LmV4YW1wbGUuaXQvaW1hZ2VzL0V1cm9wZWFuRGlzYWJpbGl0eUNhcmQvaW1hZy13YXRlcm1
hcmstSVQuc3ZnIiwNCgkJCQkJCQkJCQkidXJpI2ludGVncml0eSI6ICIzR0daSkg3aWdreWVwclhJOVBsbTNkUXNXTVlPZXV6M
E1hRUszNFBDN3NPdExmZ2VvSzlzY1hPdk1laUFDRjhUazJXT1QzNk5zWUNQZHVvTHlPRDFTZz09Ig0KDQoJCQkJCQkJCQl9DQo
JCQkJCX0sDQoJCQkJCX0sNCgkJCQkJCQkJCSJuYW1lIjogIkV1cm9wZWFuIERpc2FiaWxpdHkgQ2FyZCIsDQoJCQkJCQkibG9jY
WxlIjogImVuLVVTIiwNCgkJCQkJCQkJCSJkZXNjcmlwdGlvbiI6ICJEaWdpdGFsIHZlcnNpb24gb2YgdGhlIEV1cm9wZWFuIER

Gli elementi contenuti in `authorization_servers` nei metadati del Credential Issuer possono essere utilizzati per ottenere i metadati dell'OAuth Authorization Server tramite l'endpoint `/.well-known/oauth-authorization-server`, come definito nella Sezione 3 del **RFC 8414**. Nel caso in cui il parametro `authorization_servers` venga omesso, è possibile utilizzare l'identificativo del Credential Issuer per recuperare i metadati del Authorization Server.

Di seguito è riportato un esempio non normativo.

```
GET /.well-known/oauth-authorization-server HTTP/1.1
Host: oauth-authorization-server.example.com
```

OAuth Authorization Server DEVE rispondere con Status Code HTTP 200 e restituire i metadati dell'OAuth Authorization Server, contenenti i parametri definiti nella sezione credential-issuer-metadata:Metadata per oauth_authorization_server, all'interno di un documento JSON utilizzando il media type *application/json*.

13.2.2 Endpoint di Rilascio degli Attestati Elettronici

Pushed Authorization Request Endpoint

Pushed Authorization Request (PAR) Request

La request all'authorization endpoint del Credential Issuer DEVE contenere sia i parametri di header HTTP che i parametri HTTP POST.

Il metodo HTTP POST DEVE avere i parametri nel body del messaggio codificati in formato `application/x-www-form-urlencoded`.

Tabella 13.17: Parametri della PAR request http

Claim	Descrizione	Riferimento
client_id	DEVE essere valorizzato con il *thumbprint* del `jwk` presente nel parametro `cnf` all'interno dell'Attestato di Unità di Wallet.	**RFC 6749**
request	DEVE essere un JWT firmato. La chiave privata corrispondente a quella pubblica presente nel parametro `cnf` all'interno dell'Attestato di Unità di Wallet che DEVE essere utilizzata per firmare ilRequest Object.	OpenID Connect Core. Sezione 6

Il Pushed Authorization Endpoint è protetto con *OAuth 2.0 Attestation-based Client Authentication* [OAUTH-ATTESTATION-CLIENT-AUTH], pertanto la richiesta all'authorization endpoint del Credential Issuer DEVE utilizzare i seguenti parametri di header HTTP:

Tabella 13.18: parametri di header della request http

OAuth-Client-Attestation	DEVE contenere il JWT dell'Attestato di Unità di Wallet.	OAUTH-ATTESTATION-CLIENT-AUTH.
OAuth-Client-Attestation-PoP	DEVE contenere la Prova di Possesso del JWT dell'Attestato di Unità di Wallet.	OAUTH-ATTESTATION-CLIENT-AUTH.

Nota

I client DOVREBBERO selezionare gli algoritmi da utilizzare per la Wallet Instance Attestation e la relativa prova di possesso sulla base dei campi di metadata dell'Authorization Server `client_attestation_signing_alg_values_supported` e `client_attestation_pop_signing_alg_values_supported` documentati in credential-issuer-metadata:Metadata per oauth_authorization_server.

Il JWT *Request Object* ha i seguenti parametri di header JOSE:

JOSE Header	Descrizione	Riferimento
alg	Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	**RFC 7516# section-4.1.1**.
kid	Identificativo univoco del `jwk` presente all'interno del claim `cnf` dell'Attestato di Unità di Wallet, ovvero il valore del *thumbprint* del JWK codificato in base64url.	**RFC 7638# section_3**.
typ	OBBLIGATORIO. Media Type del JWT. DEVE essere valorizzato con `oauth-authz-req+jwt` per dichiarare esplicitamente che questo JWT è un Authorization Request Object e prevenire attacchi di *JWT confusion*.	**RFC 9101** e **RFC 8725**.

Il payload del JWT `request` contenuto nel messaggio HTTP POST contiene i seguenti parametri:

Claim	Descrizione	Riferimento
iss	DEVE essere valorizzato con il `client_id`.	**RFC 9126** e **RFC 7519**.
aud	DEVE essere valorizzato con l'identificativo del Credential Issuer.	**RFC 9126** e **RFC 7519**.

continues on next page

Tabella 13.20 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
exp	Timestamp UNIX con orario di scadenza del JWT. Il valore NON DEVE essere superiore a 300 secondi rispetto all'orario di emissione.	**RFC 9126** e **RFC 7519**.
iat	Timestamp UNIX con data e orario di emissione del JWT.	**RFC 9126** e **RFC 7519**.
response_type	DEVE essere valorizzato con `code`.	**RFC 6749**
client_id	DEVE essere valorizzato come indicato nella *Tabella dei parametri HTTP* (pagina 595).	Vedi *Tabella dei parametri HTTP* (pagina 595).
state	Identificativo univoco della sessione lato client. Questo valore verrà restituito al client nella response, al termine dell'autenticazione. DEVE essere una stringa casuale composta da caratteri alfanumerici e con una lunghezza minima di 32 cifre. Tra i caratteri speciali DEVONO essere considerati quelli non alfanumerici definiti in [NIST].	Vedi [OIDC] Sezione 3.1.2.1.
code_challenge	*Challenge* derivata dal **code verifier** che viene inviata nell'authorization request.	**RFC 7636#section-4.2**.
code_challenge_method	Metodo utilizzato per derivare il **code challenge**. DEVE essere valorizzato con `S256`.	**RFC 7636#section-4.3**.
scope	OBBLIGATORIO. Stringa JSON. Stringa contenente un identificativo univoco dell'Attestato Elettronico indipendentemente dal suo formato, come richiesto da [OPENID4VC-HAIP]. DEVE essere mappato nel claim *credential_configurations_supported* presente nei Metadata del Credential Issuer. Il valore dell'identificativo univoco DEVE corrispondere al parametro *credential_type* del *Catalogo degli Attestati Elettronici* (pagina 190). Ad esempio, nel caso del PID, può essere valorizzato con `pid`, per l'IT-Wallet ID può essere valorizzato con `eid` mentre nel caso della patente di guida `mDL`. Poiché PUÒ essere multivalore, quando ciò si verifica ogni valore DEVE essere separato da uno spazio. L'Istanza del Wallet DEVE utilizzare il valore `scope` dichiarato per la Credential Configuration richiesta, ottenuto dai Metadata del Credential Issuer o, nel flusso avviato dall'Issuer, dalla Credential Offer.	**RFC 6749** e [OPENID4VC-HAIP].

continues on next page

Tabella 13.20 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
authorization_details	OBBLIGATORIO. Array di Oggetti JSON. L'array DEVE contenere almeno un Oggetto JSON di tipo `openid_credential`. Ogni Oggetto JSON di tipo `openid_credential` DEVE includere i seguenti claim: • **type**: DEVE essere valorizzato con `openid_credential`, • **credential_configuration_id**: Stringa JSON. Stringa che indica un identificativo univoco dell'Attestato Elettronico in uno specifico formato che DEVE essere mappato nel claim *credential_configurations_supported* presente nei Metadata del Credential Issuer. Ad esempio, `dc_sd_jwt_pid` può essere utilizzato per il PID in formato SD-JWT VC, `dc_sd_jwt_eid` per l'IT-Wallet ID in formato SD-JWT VC, `dc_sd_jwt_mDL` per la patente di guida in formato SD-JWT VC e `mso_mdoc_mDL` per la patente di guida in formato mdoc. L'array PUÒ includere un Oggetto JSON aggiuntivo usato per indicare come *hint* l'Autenticazione eID Substantial con Verifica MRTD, secondo credential-issuance-endpoint: Selezione del Metodo di Autenticazione dell'Utente. Quando tale oggetto è incluso, DEVE contenere i seguenti claim: • **type**: OBBLIGATORIO. DEVE essere `it_l2+document_proof`, • **idphinting**: OPZIONALE. URL del Provider di Identità da utilizzare come *hint* opzionale per l'*Authorization Server*, secondo credential-issuance-endpoint: Selezione del Metodo di Autenticazione dell'Utente, • **challenge_method**: OBBLIGATORIO. Specifica il metodo di verifica MRTD. Il valore DEVE essere `mrtd+ias`. Metodi di verifica aggiuntivi POSSONO essere definiti in future release di questa Specifica, • **challenge_redirect_uri**: OBBLIGATORIO. Redirect URI per il processo di challenge. Il valore DEVE essere l'Universal Link registrato dall'Istanza del Wallet.	Vedi [RAR **RFC 9396**], [Open ID4VCI] e *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482).
redirect_uri	*Redirection URI* a cui è indirizzata la response. DEVE essere un *universal link* oppure un *app link* registrato nel sistema operativo locale, in modo tale che quest'ultimo potrà fornirà la response all'Istanza del Wallet.	Vedi [OIDC] Sezione 3.1.2.1.

continues on next page

Tabella 13.20 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
jti	Identificativo univoco del JWT che, insieme al valore contenuto nel claim `iss`, impedisce il riutilizzo del JWT (*replay attack*). Siccome il valore del *jti* da solo non è resistente alle collisioni, esso DEVE essere identificato in modo univoco insieme al suo emittente.	[**RFC 7519**].
issuer_state	DEVE essere presente solo in caso di issuer initiated flow. DEVE contenere lo stesso valore presente nel Credential Offer.	[**RFC 7519**].

Nota

La *Pushed Authorization Request* DEVE includere sia il parametro *scope* sia il parametro *authorization_details*. Il Credential Issuer DEVE interpretarli individualmente. Tuttavia, se entrambi richiedono lo stesso tipo di Attestato Elettronico, il Credential Issuer DEVE eseguire la richiesta come se pervenuta soltanto dall'oggetto authorization details.

Il JOSE Header della prova di possesso dell'Attestato di Unità di Wallet, contenuta negli header della HTTP Request, DEVE contenere:

JOSE Header	Descrizione	Riferimento
typ	DEVE essere valorizzato con `oauth-client-attestation-pop+jwt`	OAUTH-ATTESTATION-CLIENT-AUTH.
alg	Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	**RFC 7516# section-4.1.1**.

Il body del JWT relativo alla prova di possesso dell'Attestato di Unità di Wallet, contenuto negli header della HTTP Request, contiene:

Claim	Descrizione	Riferimento
iss	OBBLIGATORIO. *thumbprint* del JWK contenuto nel parametro `cnf`.	**RFC 9126** e **RFC 7519**.
aud	OBBLIGATORIO. DEVE essere valorizzato con l'identificativo del Credential Issuer.	**RFC 9126** e **RFC 7519**.

continues on next page

Tabella 13.22 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
iat	OBBLIGATORIO. Timestamp UNIX con data e orario di emissione del JWT.	**RFC 9126** e **RFC 7519**.
jti	OBBLIGATORIO. Identificativo univoco per il JWT *DPoP proof*. Il valore DOVREBBE essere impostato utilizzando un valore *UUID v4* secondo [**RFC 4122**].	[**RFC 7519**. Sezione 4.1.7].
nbf	OPZIONALE. Timestamp UNIX con data e orario prima del quale il JWT NON DEVE essere accettato.	**RFC 9126** e **RFC 7519**.

Pushed Authorization Request (PAR) Response

Se i controlli hanno esito positivo, il Credential Issuer DEVE fornire la response con *status code HTTP 201*. I seguenti parametri sono inclusi come parametri di primo livello nel body del messaggio di HTTP Response, utilizzando il media type `application/json` come definito nel [**RFC 8259**].

Claim	Descrizione	Riferimento
request_uri	Request URI associato all'authorization request inviata. Questo URI DEVE essere utilizzabile una sola volta per la corrispondente authorization request. DEVE contenere alcune parti generate utilizzando un algoritmo pseudocasuale crittograficamente forte. Il suo valore DEVE seguire il seguente formato: `urn:ietf:params:oauth:request_uri:<reference-value>` con `<reference-value>` come parte random dello URI che fa riferimento ai dati inviati nell'authorization request.	[**RFC 9126**].
expires_in	*JSON number*, numero intero positivo, che rappresenta la durata del Request URI in secondi.	[**RFC 9126**].

Se si verificano errori durante la PAR Request, l'Authorization Server DEVE restituire una response di errore come definito nel **RFC 9126#section-2.3**. La response DEVE avere un content type *application/json* e DEVE includere i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma *human-readable* che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una response di errore.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
```

```
{
    "error": "invalid_request",
    "error_description": "The redirect_uri is not valid for the given client"
}
```

Nella seguente tabella sono elencati gli *Status Code HTTP* e i relativi codici di errore supportati per la response di errore:

Status Code	codice di errore	Descrizione
400 Bad Request [OBBLIGATORIO]	`invalid_request`	Il Credential Issuer non può soddisfare la richiesta a causa di parametri mancanti, parametri non validi o richiesta malformata. (**RFC 6749#section-5.2**).
400 Bad Request [OBBLIGATORIO]	`invalid_scope`	Il Credential Issuer non può soddisfare la richiesta perché lo scope richiesto non è valido oppure è sconosciuto. (**RFC 6749#section-5.2**).
400 Bad Request [OBBLIGATORIO]	`use_fresh_attestation`	Il Wallet Instance Attestation JWT non è abbastanza recente per essere accettato dal server. Sezione 6.2 di OAUTH-ATTESTATION-CLIENT-AUTH.
401 Unauthorized [OBBLIGATORIO]	`invalid_client`	Il Credential Issuer non può soddisfare la richiesta a causa del fallimento della *Client Authentication* (ad esempio in caso di client sconosciuto, nessun parametro relativo alla Client Authentication presente oppure se il metodo di autenticazione non è supportato). (**RFC 6749#section-5.2**).
405 Method not allowed [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta perché il metodo POST non è stato utilizzato nella richiesta. (**RFC 9126#section-2.3**).
413 Payload Too Large [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta perché la dimensione della richiesta è superiore al limite consentito. (**RFC 9126#section-2.3**).
429 Too Many Requests [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta perché il numero di richieste ricevute è superiore al limite consentito. (**RFC 9126#section-2.3**).
500 Internal Server Error [OBBLIGATORIO]	`server_error`	Il Credential Issuer ha riscontrato un problema interno. (**RFC 6749#section-4.1.2.1**).
503 Service Unavailable [OBBLIGATORIO]	`temporarily_unavailable`	Il Credential Issuer è temporaneamente non disponibile. (**RFC 6749#section-4.1.2.1**).
504 Gateway Timeout [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta entro l'intervallo di tempo definito.

Authorization endpoint

L'authorization endpoint viene utilizzato per interagire con il Credential Issuer e ottenere un *authorization grant*. L'authorization server DEVE prima verificare l'identità dell'Utente proprietario dell'Attestato Elettronico.

Selezione del Metodo di Autenticazione dell'Utente

L'*Authorization Server* del Credential Issuer è responsabile dell'autenticazione dell'Utente secondo le proprie politiche di autenticazione e il Trust Framework applicabile. La decisione su quale metodo di autenticazione applicare, e l'eventuale disposizione di una pagina di *discovery* dei metodi supportati, DEVE restare in capo all'*Authorization Server* del Credential Issuer.

Per l'emissione di un PID:

- L'*Authorization Server* DEVE autenticare l'Utente utilizzando CieID con Livello di Garanzia Alto (CIE L3).

Per l'emissione di un IT-Wallet ID:

- L'*Authorization Server* DEVE autenticare l'Utente utilizzando uno dei metodi da esso supportati, tra cui CieID con Livello di Garanzia Alto (CIE L3) e, se supportato, l'Autenticazione eID Substantial con Verifica MRTD definita in *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482).
- L'*Authorization Server* PUÒ disporre una pagina di *discovery* che consente all'Utente di selezionare tra i metodi di autenticazione da esso supportati e ammessi per la richiesta in corso.
- L'Istanza del Wallet PUÒ includere nella *Pushed Authorization Request* uno o più *hint* opzionali relativi al metodo di autenticazione preferito dall'Utente, secondo quanto definito di seguito. Tali *hint* NON DEVONO essere interpretati come imposizione vincolante del metodo di autenticazione da parte dell'Istanza del Wallet, ma costituiscono raccomandazioni essenziali da seguire al fine di non compromettere l'Esperienza Utente.
- Quando l'Istanza del Wallet indica l'Autenticazione eID Substantial con Verifica MRTD, DEVE includere nell'array `authorization_details` un oggetto con `type` valorizzato a `it_l2+document_proof`, con la struttura e i claim definiti nella Tabella dei parametri JWT Request della Sezione credential-issuance-endpoint:Pushed Authorization Request Endpoint.
- Se l'*Authorization Server* riceve un *hint* che supporta e accetta, DOVREBBE omettere la disposizione di una pagina di *discovery* e/o l'applicazione autonoma del solo metodo di autenticazione di *default*.

- Se l'*hint* è assente, non supportato o non accettato, l'*Authorization Server* DEVE applicare le proprie politiche di autenticazione, inclusa l'eventuale pagina di *discovery* e il metodo di autenticazione di *default*.
- L'*Authorization Server* NON DEVE offrire in una pagina di *discovery*, selezionare o altrimenti applicare l'Autenticazione eID Substantial con Verifica MRTD a meno che la *Pushed Authorization Request* includa un oggetto `it_l2+document_proof` contenente i parametri tecnici obbligatori `challenge_method` e `challenge_redirect_uri`. In assenza di tali parametri l'*Authorization Server* non può completare il successivo flusso multi-step e la conferma via browser.
- Un'Istanza del Wallet NON DEVE essere obbligata a implementare una interfaccia di *discovery* del metodo di autenticazione come prerequisito di interoperabilità per l'emissione dell'IT-Wallet ID. Eventuali interfacce di pre-selezione lato Wallet (ad esempio per migliorare l'esperienza utente) RESTANO opzionali e DEVONO limitarsi a fornire *hint* all'*Authorization Server*.

Gli *hint* di autenticazione riconosciuti per l'emissione dell'IT-Wallet ID sono i seguenti:

- la presenza nell'array `authorization_details` di un oggetto con `type` valorizzato a `it_l2+document_proof` costituisce un *hint* opzionale con cui l'Istanza del Wallet indica la preferenza per l'Autenticazione eID Substantial con Verifica MRTD;
- il claim `idphinting` all'interno di tale oggetto, quando presente, costituisce un *hint* opzionale relativo all'URL del Provider di Identità LoA3 da utilizzare nel contesto di tale metodo;
- l'assenza dell'oggetto `it_l2+document_proof` NON costituisce, di per sé, un mandato dell'Istanza del Wallet sul metodo di autenticazione: in tal caso l'*Authorization Server* applica le proprie politiche e NON DEVE offrire o applicare l'Autenticazione eID Substantial con Verifica MRTD per quella richiesta. In assenza di selezione dell'Utente tramite pagina di *discovery* del Credential Issuer, il metodo di autenticazione di *default* del Provider di IT-Wallet ID DEVE essere CieID con Livello di Garanzia Alto (CIE L3).

Nota

La presente Sezione riguarda la selezione del metodo di autenticazione in sede di emissione PID e IT-Wallet ID e NON si applica alla *Discovery Page* dei Touchpoint delle Relying Party descritta in *Design dell'Esperienza Utente* (pagina 39), che ha scopo distinto (scelta del metodo di accesso a un servizio).

Per l'emissione di un (Q)EAA, l'*Authorization Server* DEVE autenticare l'Utente richiedendo la presentazione di un PID valido all'Istanza del Wallet mediante [OpenID4VP], come descritto in *Flussi Dettagliati per l'Emissione di Attestati Elettronici* (pagina 447). Il Credential Issuer

PUÒ invece richiedere la presentazione di un IT-Wallet ID valido, secondo le proprie politiche di autenticazione. In tale contesto NON si applica la *discovery* dei metodi SPID, CIE L3 o L2+.

Authorization Request

L'authorization request viene inviata dal Browser Web in uso dall'Istanza del Wallet usando i metodi HTTP **POST** o **GET**. Se viene utilizzato il metodo **POST**, i parametri DEVONO essere inviati utilizzando la *Form Serialization*. Quando viene utilizzato il metodo **GET**, i parametri DEVONO essere inviati utilizzando la *Query String Serialization*. Per maggiori dettagli vedere la Sezione 13 di [OIDC].

I parametri obbligatori nell'authentication request HTTP sono specificati nella seguente tabella.

Claim	Descrizione	Riferimento
client_id	DEVE essere valorizzato come indicato nella *Tabella dei parametri HTTP* (pagina 595).	Vedi *Tabella dei parametri HTTP* (pagina 595).
request_uri	DEVE essere valorizzato con lo stesso valore ottenuto dalla PAR Response. Vedi *Tabella dei parametri della Risposta HTTP PAR* (pagina 600).	[**RFC 9126**].

Authorization Response

L'authentication response viene restituita dall'authorization endpoint del Credential Issuer al termine del flusso di autenticazione.

Se l'autenticazione ha esito positivo, il Credential Issuer reindirizza l'Utente aggiungendo i seguenti parametri di query al *redirect_uri*. Il redirect URI DEVE essere un *universal link* o un *app link* registrato nel sistema operativo locale, in modo che quest'ultimo sia in grado di fornire la response all'Istanza del Wallet.

Claim	Descrizione	Riferimento
code	*Authorization Code* univoco che l'Istanza del Wallet invia al Token Endpoint.	[**RFC 6749# section-4.1.2**], [**RFC 7521**].
state	L'Istanza del Wallet DEVE verificare la corrispondenza con il valore presente nel parametro `state` del Request Object, come definito nella *Tabella dei parametri della Richiesta JWT* (pagina 596).	[**RFC 6749# section-4.1.2**].

continues on next page

Tabella 13.26 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
iss	Identificativo univoco del Credential Issuer che ha creato l'Authentication Response. L'Istanza del Wallet DEVE validare questo parametro.	[RFC 9207], [RFC 7519, Sezione 4.1.1.].

Se si verificano errori durante l'Authorization Request, l'Authorization Server DEVE restituire una response di errore come definito nel RFC 6749#section-4.1.2.1. In caso di `redirect_uri` o `client_id` non valido/mancante, l'Authorization Server DEVE informare l'Utente con l'errore e NON DEVE reindirizzare l'Utente verso il *Redirection URI*. Se si verifica qualsiasi altro errore, l'Authorization Server DEVE reindirizzare l'Utente aggiungendo i seguenti parametri in query come richiesto al *redirect_uri* utilizzando il formato *application/x-www-form-urlencoded*:

- *error*. Il codice di errore.
- *error_description*. Testo in forma *human-readable* che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.
- *state*. Il valore esatto del parametro `state` contenuto nel Request Object.

Di seguito è riportato un esempio non normativo di una response di errore.

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?
 error=invalid_request
 &error_description=Unsupported%20response_type%20value
 &state=fyZiOL9Lf2CeKuNT2JzxiLRDink0uPcd
```

Nel caso in cui l'Authorization Server reindirizza l'Utente verso il *redirect_uri* DEVE essere utilizzato lo *status code HTTP 302 (Found)*. I seguenti codici di errore sono supportati per la response di errore:

Status Code	codice di errore	Descrizione
302 Found [OBBLIGATORIO]	`invalid_request`	Il Credential Issuer non può soddisfare la richiesta a causa di parametri mancanti, parametri non validi o richiesta malformata. (RFC 6749#section-4.1.2.1).
302 Found [OBBLIGATORIO]	`unauthorized_client`	Il Credential Issuer non può soddisfare la richiesta perché il client non è autorizzato a richiedere *authorization code*. (RFC 6749#section-4.1.2.1).
302 Found [OBBLIGATORIO]	`server_error`	Il Credential Issuer ha riscontrato un problema interno. (RFC 6749#section-4.1.2.1).
302 Found [OBBLIGATORIO]	`temporarily_unavailable`	Il Credential Issuer è temporaneamente non disponibile. (RFC 6749#section-4.1.2.1).

Nel caso in cui l'Authorization Server non reindirizza l'Utente verso il *redirect_uri* i seguenti *Status Code HTTP* sono supportati per la respomse di errore:

Codice di Stato	Descrizione
400 Bad Request [OBBLIGATORIO]	Il Credential Issuer non può soddisfare la richiesta a causa del parametro `redirect_uri` o `client_id` non valido/mancante.
500 Internal Server Error [OBBLIGATORIO]	Il Credential Issuer ha riscontrato un problema interno.
503 Service Unavailable [OBBLIGATORIO]	Il Credential Issuer è temporaneamente non disponibile.
504 Gateway Timeout [OPZIONALE]	Il Credential Issuer non può soddisfare la richiesta entro l'intervallo di tempo definito.

Token Endpoint

Il Token Endpoint viene utilizzato dall'Istanza del Wallet per ottenere un *Access Token* previa presentazione dell'*authorization grant*, come definito nel **RFC 6749**. Il Token Endpoint è un endpoint protetto con *OAuth 2.0 Attestation-based Client Authentication* [OAUTH-ATTESTATION-CLIENT-AUTH].

Token Request

La richiesta al Token Endpoint del Credential Issuer DEVE essere una HTTP request con metodo POST, con il body del messaggio codificato in formato `application/x-www-form-urlencoded`. L'Istanza del Wallet invia la richiesta al Token Endpoint con `OAuth-Client-Attestation` e `OAuth-Client-Attestation-PoP` come parametri di header secondo OAUTH-ATTESTATION-CLIENT-AUTH.

Il Token Endpoint è protetto con *OAuth 2.0 Attestation-based Client Authentication* [OAUTH-ATTESTATION-CLIENT-AUTH], pertanto la richiesta all'authorization endpoint del Credential Issuer DEVE utilizzare i seguenti parametri di header HTTP **OAuth-Client-Attestation** e **OAuth-Client-Attestation-PoP** come definito in credential-issuance-endpoint:Pushed Authorization Request Endpoint.

Il Token Endpoint emette il token DPoP, pertanto è OBBLIGATORIO che la request includa nel suo header HTTP il parametro *DPoP proof*. L'Authorization Server DEVE convalidare il

DPoP proof ricevuto al Token Endpoint, secondo quanto indicato nella Sezione 4.3 di **RFC 9449**. Ciò mitiga l'uso improprio di Access Token/Refresh Token persi o rubati al Credential/ Token Endpoint. Se il *DPoP proof* non è valido, il Token Endpoint restituisce una response di errore, secondo quanto deinito nella Sezione 5.2 del [**RFC 6749**] con `invalid_dpop_proof` come valore del parametro di errore.

La token request contiene i seguenti claim:

Claim	Descrizione	Riferimento
grant_type	OBBLIGATORIO. DEVE essere valorizzato con `authorization_code` o `refresh_token`.	[**RFC 6749**].
code	OBBLIGATORIO solo se il *grant type* è `authorization_code`. L'Authorization code restituito nell'Authentication Response. NON DEVE essere presente se il *grant type* è `refresh_token`.	[**RFC 6749**].
redirect_uri	OBBLIGATORIO solo se il *grant type* è `authorization_code`. DEVE essere valorizzato come nel Request Object *Tabella dei parametri della Richiesta JWT* (pagina 596). NON DEVE essere presente se il *grant type* è `refresh_token`.	[**RFC 67491**].
code_verifier	OBBLIGATORIO solo se il *grant type* è `authorization_code`. Codice di verifica del **code_challenge**.	Proof Key for Code Exchange by OAuth Public Clients. NON DEVE essere presente se il *grant type* è `refresh_token`.
refresh_token	OBBLIGATORIO solo se il *grant type* è `refresh_token`. Il Refresh Token precedentemente emesso all'Istanza del Wallet. NON DEVE essere presente se il *grant type* è `authorization_code`.	[**RFC 6749**].
scope	OPZIONALE solo se il *grant type* è `refresh_token`. Lo scope richiesto NON DEVE includere alcun valore di scope non originariamente concesso dall'Utente, e se omesso è da intendersi uguale allo scope originariamente concesso dall'Utente. NON DEVE essere presente se il *grant type* è `authorization_code`.	[**RFC 6749**].

Un *JWT DPoP Proof* è incluso nella request HTTP utilizzando il parametro di header `DPoP` contenente un JWT DPoP.

Il JOSE Header di un **JWT DPoP** DEVE contenere almeno i seguenti parametri:

JOSE Header	Descrizione	Riferimento
typ	DEVE essere uguale a `dpop+jwt`.	[**RFC 7515**] e [**RFC 8725**. Sezione 3.11].
alg	Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	[**RFC 7515**].
jwk	Rappresenta la chiave pubblica scelta dall'Istanza del Wallet, in formato JSON Web Key (JWK) [**RFC 7517**] a cui l'Access Token DEVE essere vincolato, come definito nella Sezione 4.1. 3 del [**RFC 7515**]. NON DEVE contenere una chiave privata.	[**RFC 7517**] e [**RFC 7515**].

Il payload del **JWT DPoP Proof** DEVE contenere i seguenti claim:

Claim	Descrizione	Riferimento
jti	Identificativo univoco per il JWT *DPoP proof*. Il valore DOVREBBE essere un valore *UUID v4* secondo [**RFC 4122**].	[**RFC 7519**. Sezione 4.1.7].
htm	Il valore del metodo HTTP della request a cui è allegato il JWT.	[**RFC 9110**. Sezione 9.1].
htu	Target URI HTTP, senza le parti di query e fragment, della request a cui è allegato il JWT.	[**RFC 9110**. Sezione 7.1].
iat	Timestamp UNIX con data e orario di emissione del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[**RFC 7519**. Sezione 4.1.6].

Token Response

Se la Token Request viene validata con successo, l'Authorization Server fornisce una Token Response con *status code HTTP 200 (OK)*. La Token Response contiene i seguenti claim.

Claim	Descrizione	Riferimento
access_token	OBBLIGATORIO. Il *DPoP-bound Access Token*, in formato JWT firmato, consente di accedere al Credential Endpoint per ottenere l'Attestato Elettronico.	[**RFC 6749**].
refresh_token	OPZIONALE. Il *DPoP-bound Refresh Token*, in formato JWT firmato, che può essere utilizzato per ottenere un nuovo Access Token presso il Token Endpoint del Credential Issuer.	[**RFC 6749**].
token_type	OBBLIGATORIO. Tipo di *Access Token* restituito. DEVE essere uguale a `DPoP`.	[**RFC 6749**].

continues on next page

Tabella 13.32 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
expires_in	OBBLIGATORIO. Tempo di scadenza dell'*Access Token* in secondi.	[**RFC 6749**].
authorization_details	OBBLIGATORIO. Array di Oggetti JSON, utilizzati per identificare gli Attestati Elettronici con gli stessi Metadata ma diversi valori di claimset/claim e/o semplificare la Credential Request anche quando viene emessa un solo Attestato Elettronico. Oltre al claim definito nella *Tabella dei parametri della Richiesta JWT* (pagina 596) DEVE includere il seguente claim: • **credential_identifiers**: Array di stringhe, ciascuna che identifica in modo univoco un set di dati dell'Attestato Elettronico disponibile per l'emissione.	[OpenID4VCI].

Se si verificano errori durante la convalida della Token Request, l'Authorization Server DEVE restituire una response di errore come definito nel **RFC 6749#section-5.2**. La response DEVE utilizzare il Content-Type HTTP *application/json* e DEVE includere i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma *human-readable* che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una response di errore.

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "error": "invalid_client",
    "error_description": "Client authentication failed"
}
```

Nella seguente tabella sono elencati i *status code HTTP* e i relativi codici di errore supportati per la response di errore:

Status Code	codice di errore	Descrizione
400 Bad Request [OBBLIGATORIO]	`invalid_request`	Il Credential Issuer non può soddisfare la richiesta a causa di parametri mancanti, parametri non validi o richiesta malformata. (**RFC 6749#section-5.2**).

continues on next page

Tabella 13.33 – continua dalla pagina precedente

Status Code	codice di errore	Descrizione
400 Bad Request [OBBLIGATORIO]	`invalid_grant`	Il Credential Issuer non può soddisfare la richiesta perché l'*authorization code* o il *Refresh Token* fornito non è valido, è scaduto, è stato revocato o non corrisponde al *redirection URI* utilizzato nell'authorization request, o è stato rilasciato ad un altro client. (**RFC 6749#section-5.2**).
400 Bad Request [OBBLIGATORIO]	`unsupported_grant_type`	Il Credential Issuer non può soddisfare la richiesta perché il *grant type* non è supportato. (**RFC 6749#section-5.2**).
400 Bad Request [OBBLIGATORIO]	`invalid_dpop_proof`	Il Credential Issuer non può soddisfare la richiesta a causa di un *DPoP proof* non valido. Sezione 5 del [**RFC 9449**].
400 Bad Request [OBBLIGATORIO]	`use_fresh_attestation`	Il Wallet Instance Attestation JWT non è abbastanza recente per essere accettato dal server. Sezione 6.2 di OAUTH-ATTESTATION-CLIENT-AUTH.
401 Unauthorized [OBBLIGATORIO]	`invalid_client`	Il Credential Issuer non può soddisfare la richiesta a causa del fallimento della *Client Authentication* (ad esempio in caso di client sconosciuto, nessun parametro relativo alla Client Authentication presente oppure se il metodo di autenticazione non è supportato). (**RFC 6749#section-5.2**).
500 Internal Server Error [OBBLIGATORIO]	`server_error`	Il Credential Issuer ha riscontrato un problema interno.
503 Service Unavailable [OBBLIGATORIO]	`temporarily_unavailable`	Il Credential Issuer è temporaneamente non disponibile.
504 Gateway Timeout [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta entro l'intervallo di tempo definito.

Access Token

Un DPoP-bound Access Token viene fornito dal Token Endpoint del Credential Issuer come risultato di una token request andata a buon fine. L'Access Token è codificato in formato JWT, secondo [**RFC 7519**]. L'Access Token DEVE avere almeno i seguenti claim obbligatori e DEVE essere vincolato alla chiave pubblica fornita dal *DPoP proof*. Questo vincolo può essere realizzato in base alla metodologia definita nella Sezione 6 del (**RFC 9449**).

Il **JWT DPoP** contiene i seguenti parametri di header JOSE e claim.

JOSE Header	Descrizione	Riferimento
typ	OBBLIGATORIO. DEVE essere uguale a `at+jwt`.	[**RFC 7515**].
alg	OBBLIGATORIO. Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	[**RFC 7515**].
kid	OBBLIGATORIO. Identificativo univoco del `jwk` utilizzato dal Credential Issuer per firmare l'Access Token.	**RFC 7638# section_3**.

Claim	Descrizione	Riferimento
iss	OBBLIGATORIO. DEVE essere un URL HTTPS che identifica in modo univoco il Credential Issuer. L'Istanza del Wallet DEVE verificare che questo valore corrisponda al Credential Issuer a cui ha richiesto l'Attestato Elettronico.	[**RFC 9068**], [**RFC 7519**].
sub	OBBLIGATORIO. Identifica il soggetto del JWT.	[**RFC 9068**], [**RFC 7519**] e Sezione 8 di [OIDC].
client_id	OBBLIGATORIO. L'identificativo dell'Istanza del Wallet che ha richiesto l'Access Token; DEVE essere uguale al kid della chiave pubblica dell'Istanza del Wallet specificata nell'Attestato di Unità di Wallet (`cnf.jwk`).	[**RFC 9068**], [**RFC 7519**] e Sezione 8 di [OIDC].
aud	OBBLIGATORIO. DEVE essere valorizzato con l'identificativo del Credential Issuer.	[**RFC 9068**].
iat	OBBLIGATORIO. Timestamp UNIX con data e orario di emissione del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[**RFC 9068**], [**RFC 7519**. Sezione 4.1.6].
exp	OBBLIGATORIO. Timestamp UNIX con data e orario di scadenza del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[**RFC 9068**], [**RFC 7519**].
jti	OPZIONALE. DEVE essere una Stringa in formato *uuid4*. Identificativo univoco del Token ID che la RP DOVREBBE utilizzare per prevenire il riutilizzo rifiutando l'ID Token se è stato già elaborato.	[**RFC 9068**], [**RFC 7519**].
cnf	OBBLIGATORIO. DEVE contenere un claim **jkt** che è un Metodo di Conferma del *Thumbprint JWK SHA-256*. Il valore del parametro *jkt* DEVE contenere la codifica base64url (come definito nel [**RFC 7515**]) del *Thumbprint JWK SHA-256* della chiave pubblica DPoP (in formato JWK) a cui è vincolato l'Access Token.	[**RFC 9449**. Sezione 6.1] e [**RFC 7638**].

Refresh Token

Un *DPoP-bound Refresh Token* viene fornito dal Token endpoint del Credential Issuer come risultato di una token request andata a buon fine. Il Refresh Token è codificato in formato JWT, secondo [**RFC 7519**]. Il Refresh Token DEVE avere almeno i seguenti claim obbligatori e DEVE essere vincolato alla chiave pubblica fornita dal *DPoP proof*. Questo vincolo può essere realizzato in base alla metodologia definita nella Sezione 6 del (**RFC 9449**).

Il **JWT DPoP** DEVE contenere i seguenti parametri di header JOSE e claim.

JOSE Header	Descrizione	Riferimento
typ	DEVE essere uguale a `rt+jwt`.	[**RFC 7515**].
alg	Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	[**RFC 7515**].
kid	Identificativo univoco del `jwk` utilizzato dal Credential Issuer per firmare l'Access Token.	**RFC 7638# section_3**.

Claim	Descrizione	Riferimento
iss	DEVE essere un URL HTTPS che identifica in modo univoco il Credential Issuer. L'Istanza del Wallet DEVE verificare che questo valore corrisponda al Credential Issuer a cui ha richiesto l'Attestato Elettronico.	[**RFC 9068**], [**RFC 7519**].
sub	Identifica il soggetto del JWT. DEVE essere settato con il valore del campo `sub` presente nell'Attestato ELettronico in formato SD-JWT-VC.	[**RFC 9068**], [**RFC 7519**] e Sezione 8 di [OIDC].
client_id	L'identificativo per l'Istanza del Wallet che ha richiesto l'Access Token; DEVE essere uguale al valore del *kid* che identifica la chiave pubblica utilizzata nell'Istanza del Wallet, utilizzata nell'Attestato di Unità di Wallet (`cnf.jwk`).	[**RFC 9068**], [**RFC 7519**] e Sezione 8 di [OIDC].
aud	DEVE essere valorizzato con l'identificativo del Credential Issuer.	[**RFC 9068**].
iat	Timestamp UNIX con data e orario di emissione del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[**RFC 9068**], [**RFC 7519**. Sezione 4.1.6].

continues on next page

Tabella 13.37 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
nbf	Timestamp UNIX con data e orario prima del quale il JWT NON DEVE essere accettato, codificato come NumericDate come indicato nel **RFC 7519**. DOVREBBE essere impostato sul claim `exp` del corrispondente Token di Accesso.	[**RFC 7519**. Sezione 4.1.7].
exp	Timestamp UNIX con data e orario di scadenza del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[**RFC 9068**], [**RFC 7519**].
jti	DEVE essere una Stringa in formato *uuid4*. Identificativo univoco del Token ID che la RP DOVREBBE utilizzare per prevenire il riutilizzo rifiutando l'ID Token se è stato già elaborato.	[**RFC 9068**], [**RFC 7519**].
cnf	DEVE contenere un claim **jkt** che è un Metodo di Conferma del *Thumbprint JWK SHA-256*. Il valore del parametro *jkt* DEVE contenere la codifica base64url (come definito nel [**RFC 7515**]) del *Thumbprint JWK SHA-256* della chiave pubblica DPoP (in formato JWK) a cui è vincolato l'Access Token.	[**RFC 9449**. Sezione 6.1] e [**RFC 7638**].

Nonce Endpoint

Il Nonce Endpoint fornisce un valore del `c_nonce` utile per creare una prova di possesso del materiale crittografico per la richiesta al Credential Endpoint, come definito nella Sezione 7 di OpenID4VCI.

Nonce Request

La Nonce Request DEVE essere una HTTP POST request senza body indirizzata al Nonce Endpoint del Credential Issuer censito nei Metadata del Credential Issuer.

Nonce Response

La Nonce Response DEVE essere inviata all'Istanza del Wallet utilizzando il media type *application/json*. In caso di Nonce Request andata a buon fine, il Credential Issuer DEVE restituire una HTTP response con *status code HTTP 200 (OK)*.

Come definito nella Sezione 7.2 di OpenID4VCI, il Credential Issuer DEVE rendere la risposta non memorizzabile nella cache aggiungendo il campo di header `Cache-Control` valorizzato con *no-store*.

La Nonce Response contiene il seguente parametro:

Claim	Descrizione	Riferimento
c_nonce	OBBLIGATORIO. Stringa contenente il valore del nonce. Questo valore DEVE essere imprevedibile.	Sezione 7.2 di [OpenID4VCI].

Credential Endpoint

Il Credential Endpoint emette un'Attestato Eletronico previa presentazione di un Access Token valido, come definito in OpenID4VCI.

Credential Request

L'Istanza del Wallet quando richiede l'Attestato Elettronico al Credential Endpoint, DEVE utilizzare i seguenti parametri nel body del messaggio della HTTP POST request, utilizzando il tipo di media *application/json*.

Il Credential Endpoint DEVE accettare e convalidare il *DPoP proof* inviato nel parametro di header HTTP *DPoP*, secondo i passaggi definiti nella Sezione 4.3 del (**RFC 9449**). Il *DPoP proof* oltre ai valori definiti nella sezione Token Endpoint DEVE contenere il seguente claim:

- **ath**: valore di hash dell'Access Token codificato in ASCII. Il valore DEVE utilizzare la codifica base64url (come definito nella Sezione 2 del (**RFC 7515**) con l'algoritmo SHA-256.

Avvertimento

L'Istanza del Wallet DEVE creare un **nuovo DPoP proof** per la Credential Request e NON DEVE utilizzare la *proof* precedentemente creata per il Token Endpoint.

Claim	Descrizione	Riferimento
credential_identifier	OBBLIGATORIO quando Authorization Details di tipo *openid_credential* è stato restituito dalla Token. In tutti gli altri casi NON DEVE essere utilizzato. Questo DEVE essere valorizzato con uno dei valori ottenuti nel claim `credential_identifiers` della Token Response. NON DEVE essere utilizzato se è presente `credential_configuration_id`.	Sezione 8.2 di [OpenID4VCI].

continues on next page

Tabella 13.38 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
credential_ configuration_ id	OBBLIGATORIO se il parametro `credential_identifiers` è assente nella Token Response. In tutti gli altri casi NON DEVE essere utilizzato. Stringa che specifica un identificativo univoco dell'Attestato ELetronico descritto nel claim *credential_configurations_supported* presente nei Metadata del Credential Issuer. Ad esempio, nel caso del PID, può essere valorizzato con `pid`.	Sezione 8.2 di [OpenID4VCI].
proofs	OBBLIGATORIO. Oggetto che fornisce una o più prove di possesso del materiale crittografico a cui saranno vincolate le Istanze di Credenziali emesse. L'oggetto `proofs` DEVE contenere un parametro denominato *jwt* contenente un array di JWT, dove ogni elemento viene utilizzato come prova di possesso.	[OpenID4VCI].

Il *proof type* del JWT DEVE contenere i seguenti parametri per l'header JOSE e il body in JWT:

JOSE Header	Descrizione	Riferimento
alg	Identificativo dell'algoritmo di firma digitale come definito nel registro IANA "JSON Web Signature and Encryption Algorithms". DEVE essere uno degli algoritmi supportati elencati nella Sezione *Algoritmi Crittografici* (pagina 647) e NON DEVE essere valorizzato con `none` o con qualsiasi identificativo di algoritmo simmetrico (MAC).	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
typ	DEVE essere valorizzato con *openid4vci-proof+jwt*.	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
jwk	Rappresenta la chiave pubblica scelta dall'Istanza del Wallet, in formato JSON Web Key (JWK) [**RFC 7517**] a cui l'Attestato Elettronico sarà vincolato, come definito nella Sezione 4.1.3 del [**RFC 7515**].	[OpenID4VCI], [**RFC 7515**], [**RFC 7517**].
key_attestation	Representa la Key Attestation.	[OpenID4VCI].

Claim	Descrizione	Riferimento
iss	Il valore di questo claim DEVE essere il **client_id** dell'Istanza del Wallet.	[OpenID4VCI], [**RFC 7519**, Sezione 4.1.1].
aud	DEVE essere valorizzato con l'identificativo del Credential Issuer.	[OpenID4VCI].

continues on next page

Tabella 13.40 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
iat	Timestamp UNIX con data e orario di emissione del JWT, codificato come NumericDate come indicato nel **RFC 7519**.	[OpenID4VCI], [**RFC 7519**. Sezione 4.1.6].
nonce	Il tipo di valore di questo claim DEVE essere una stringa, dove il valore è un **c_nonce** fornito dal Credential Issuer tramite la Nonce Response.	[OpenID4VCI].

Credential Response

La Credential Response DEVE essere inviata all'Istanza del Wallet utilizzando il media type *application/json*. Se la Credential Request viene validata con successo e l'Attestato Elettronico è immediatamente disponibile, il Credential Issuer DEVE restituire una HTTP response con un *status code HTTP 200 (OK)*. Se l'Attestato Elettronico non è disponibile e il Deferred Flow è supportato dal Credential Issuer, allora DEVE essere restituito un *status code HTTP 202*.

La Credential Response contiene i seguenti parametri:

Claim	Descrizione	Riferimento
credentials	OBBLIGATORIO se `interval` e `transaction_id` non sono presenti, altrimenti NON DEVE essere presente. Array di una o più Credenziali emesse. Il numero di elementi nell'array delle Credenziali corrisponde al numero di chiavi che l'Istanza del Wallet ha fornito tramite il parametro `proofs` nella Credential Request. L'array DEVE contenere oggetti JSON, dove ogni oggetto DEVE avere il claim `credential`. Il valore del claim `credential` DEVE essere configurato con una stringa contenente la Credenziale codificata, come descritto di seguito: • **credential**: OBBLIGATORIO. Stringa contenente un Attestato Elettronico emesso. Se l'identificativo del formato richiesto è `dc+sd-jwt` allora il parametro `credential` NON DEVE essere ricodificato. Se l'identificativo di formato richiesto è `mso_mdoc` allora il parametro `credential` DEVE essere una rappresentazione codificata in base64url della struttura IssuerSigned codificata in CBOR, come definito in [ISO 18013-5]. Questa struttura DOVREBBE contenere tutti i Namespaces e IssuerSignedItems inclusi negli AuthorizedNamespaces del MobileSecurityObject.	Sezione 8.3, Allegato A2.4 e Allegato A3.4 di [OpenID4VCI].

continues on next page

Tabella 13.41 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
interval	OBBLIGATORIO se `transaction_id` è presente, altrimenti NON DEVE essere presente. La quantità di tempo (espressa in secondi) richiesta prima di effettuare una Deferred Credential Request (*WP_065* (pagina 790)).	Sezione 8.3 di [OpenID4VCI].
notification_id	OPZIONALE. Stringa che identifica un'Attestato Elettronico emesso che il Wallet include nella Notification Request come definito nella Sezione credential-issuance-endpoint: Notification Request. NON DEVE essere presente se il parametro `credentials` non è presente.	Sezione 8.3 di [OpenID4VCI].
transaction_id	OBBLIGATORIO se `credentials` non è presente, altrimenti NON DEVE essere presente. Stringa che identifica una transazione di emissione posticipata che il Wallet include nella successiva Credential Request come definito nella Sezione credential-issuance-endpoint:Deferred Endpoint come specificato in (*WP_065* (pagina 790)). DEVE essere invalidato dopo che l'Utente ottiene l'Attestato Elettronico.	Sezione 8.3 di [OpenID4VCI].

Nel caso in cui la Credential Request non contenga un Access Token valido, il Credential Endpoint restituisce una response di errore come definito nella Sezione 3 del [**RFC 6750**]. Se si verifica qualsiasi altro errore, il Credential Issuer DEVE restituire una response di errore come definito nella Sezione 8.3.1 di [OpenID4VCI]. La response DEVE utilizzare il content type *application/json* e DEVE includere i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma *human-readable* che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una response di errore.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
    "error": "invalid_proof",
    "error_description": "The proof field is not present or the provided key proof is invalid o
r not bound to a nonce provided by the Credential Issuer."
}
```

Nella seguente tabella sono elencati i *Status Code HTTP* e i relativi codici di errore supportati per la response di errore:

Status Code	codice di errore	Descrizione
400 Bad Request [OBBLIGATORIO]	`invalid_credential_request`	Il Credential Issuer non può soddisfare la richiesta a causa di parametri mancanti, parametri non validi o richiesta malformata. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`unknown_credential_configuration`	Il Credential Issuer non può soddisfare la richiesta perché il `credential_configuration_id` relativo all' Attestato Elettronico richiesto è sconosciuto. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`unknown_credential_identifier`	Il Credential Issuer non può soddisfare la richiesta perché il `credential_identifier` dell'Attestato Elettronico richiesto non è supportato. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_proof`	Il Credential Issuer non può soddisfare la richiesta perché il parametro `proofs` nella Credential Request non è valido o è assente oppure non contiene il valore del `c_nonce`. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_nonce`	Il Credential Issuer non può soddisfare la richiesta perché il parametro `proofs` nella Credential Request utilizza un nonce non valido. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_encryption_parameters`	Il Credential Issuer non può soddisfare la richiesta perché i parametri di crittografia nella Credential Request non sono validi o mancanti. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`credential_request_denied`	La Credential Request non è stata accettata dal Credential Issuer. Il Wallet DOVREBBE trattare questo errore come definitivo, il che significa che, se ricevuto da un Credential Issuer, le Credenziali non potranno essere emesse. Sezione 8.3.1 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_transaction_id`	Solo in caso di Deferred Flow. Il Credential Issuer non può soddisfare la richiesta perché la Credential Request contiene un `transaction_id` non valido. Sezione 9.3 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_dpop_proof`	Il Credential Issuer non può soddisfare la richiesta a causa del *DPoP proof* non valido. Sezione 7 del [**RFC 9449**].
500 Internal Server Error [OBBLIGATORIO]	`server_error`	Il Credential Issuer ha riscontrato un problema interno.
503 Service Unavailable [OBBLIGATORIO]	`temporarily_unavailable`	Il Credential Issuer è temporaneamente non disponibile.
504 Gateway Timeout [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta entro l'intervallo di tempo definito.

Deferred Endpoint

I Credential Issuer POSSONO supportare il *Deferred Endpoint* con l'obiettivo di soddisfare i casi in cui un'emissione immediata potrebbe non essere possibile, a causa di errori durante la comunicazione tra il Credential Issuer e la Fonte Autentica (ad esempio la Fonte Autentica è temporaneamente non disponibile, ecc.) o a causa di processi amministrativi o tecnici da espletare.

Nel caso in cui la Fonte Autentica e il Credential Issuer siano entrambi abilitati a utilizzare *PDND*, si DEVE seguire quanto descritto nella Sezione *Fonti Autentiche* (pagina 376).

Si applicano i seguenti requisiti:

1. La Deferred Credential Request PUÒ avvenire anche diversi giorni dopo l'iniziale Credenziale Request.
2. L'Utente DEVE essere informato che l'Attestato Elettronico è disponibile e pronto per essere emesso.
3. Il Fornitore di Wallet NON DEVE essere informato su quale Attestato Elettronico è disponibile per l'emissione o quale Credential Issuer l'Utente deve contattare.
4. L'Istanza del Wallet DEVE essere informata sulla quantità di tempo da attendere prima di effettuare una nuova Credential Request.
5. Poiché, in generale, un'indisponibilità può essere un evento imprevisto, il Credential Issuer DEVE essere in grado di passare al volo tra un flusso *immediato* e uno *posticipato*. Questa decisione DEVE essere presa dopo la fase autorizzativa.

Se i Credential Issuer, che supportano questo flusso, non sono in grado di emettere immediatamente un Attestato ELettronico richiesto, DEVONO fornire all'Istanza del Wallet una Credential Response HTTP contenente la quantità di tempo da attendere prima di effettuare una nuova Credential Request e un identificativo della transazione di emissione posticipata (*transaction_id*). Lo *status code HTTP* previsto DEVE essere il *202* (vedi Sezione 15.3.3 del [**RFC 9110**]). Di seguito viene fornito un esempio non normativo.

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Cache-Control: no-store

{
    "transaction_id": "8xLOxBtZp8",
    "interval": 864000
}
```

L'Istanza del Wallet DEVE utilizzare il valore fornito nel parametro *interval* per informare l'Utente quando l'Attestato Elettronico diventa disponibile (ad esempio utilizzando una notifica locale innescata dal valore di tempo *interval*). I Credential Issuer POSSONO inviare una notifica

all'Utente tramite un canale di comunicazione (ad esempio indirizzo email), se precedentemente fornito dall'Utente al Credential Issuer.

Deferred Request

Una volta ricevuta la notifica (dall'Istanza del Wallet e/o dal Credential Issuer), l'Utente accede all'Istanza del Wallet.

L'Istanza del Wallet DEVE presentare al Deferred Endpoint un Access Token valido per l'emissione dell'Attestato Elettronico precedentemente richiesto al Credential Endpoint.

Se il valore del parametro `interval` risulta inferiore rispetto alla scadenza dell'Access Token, l'Istanza del Wallet DOVREBBE utilizzare l'Access Token (*WP_066b* (pagina 790)). Altrimenti, l'Istanza del Wallet PUÒ ottenere un nuovo Access Token seguendo il flusso relativo al Refresh Token (vedi Sezione *Refresh Token Flow* (pagina 469) per maggiori dettagli come specificato in *WP_066c* (pagina 790)). Se il flusso del Refresh Token fallisce, l'Istanza del Wallet deve inviare una nuova authentication request (*WP_067* (pagina 790)).

La Deferred Credential Request DEVE essere una HTTP POST request. DEVE essere inviata utilizzando il media type `application/json`. Il seguente parametro viene utilizzato nella Deferred Credential Request:

- `transaction_id`: OBBLIGATORIO. Stringa che identifica una transazione di Emissione posticipata (*WP_066a* (pagina 790)).

Il Credential Issuer DEVE invalidare il `transaction_id` dopo che l'Attestato Elettronico per cui era destinato è stata ottenuto dall'Istanza del Wallet. Di seguito è riportato un esempio non normativo di una Deferred Credential Request:

```
POST /credential HTTP/1.1
Host: eaa-provider.example.org
Content-Type: application/json
Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
    VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
    nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R
    1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
    oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
    WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF
    c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E
    OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA


{
  "transaction_id": "8xLOxBtZp8"
}
```

Deferred Response

La Deferred Credential Response DEVE essere inviata utilizzando il media type `application/json`.

Se l'Attestato Elettronico è disponibile, la Deferred Credential Response DEVE utilizzare i parametri `credentials` e `notification_id` come definito nella Sezione credential-issuance-endpoint:Credential Response e DEVE rispondere con un HTTP Status 200 (vedi sezione 15.3.3 del **RFC 9110**).

Se il Credential Issuer ha bisogno di più tempo, la Deferred Credential Response DEVE utilizzare i parametri `interval` e `transaction_id` come definiti nella sezione credential-issuance-endpoint: Credential Response e DEVE rispondere con HTTP Status 202 (vedi Sezione 15.3.3 del **RFC 9110**).

Il valore di `transaction_id` DEVE essere identico a quello presente nella Deferred Credential Request.

Se la Deferred Credential Request non è valida o l'Attestato Elettronico non è disponibile, la Deferred Credential Error Response DEVE essere inviata all'Istanza del Wallet secondo quanto indicato nella Sezione credential-issuance-endpoint:Credential Response.

Notification Endpoint

Il Notification Endpoint viene utilizzato dal Wallet per notificare al Credential Issuer determinati eventi relativi agli Attestati Elettronici emessi, come ad esempio se l'Attestato Elettronico è stato memorizzato con successo nell'Istanza del Wallet.

Per salvaguardare la privacy, l'`event_description` nella notifica NON DOVREBBE contenere alcuna informazione che potrebbe rivelare il comportamento dell'Utente o rivelare lo stato del dispositivo personale (ad esempio, se lo spazio di archiviazione è pieno).

Questo endpoint DEVE essere protetto utilizzando un Access Token di tipo DPoP. Il protocollo TLS per la riservatezza del trasporto su HTTP è OBBLIGATORIO secondo la Sezione 11 di [OpenID4VCI].

Notification Request

La Notification Request DEVE essere una HTTP POST utilizzando il media type *application/json* con i seguenti parametri.

Claim	Descrizione	Riferimento
notification_id	OBBLIGATORIO. DEVE essere uguale al valore `notification_id` restituito nella Credential Response dal Credential Issuer.	Sezione 11.1 di [Open ID4VCI].
event	OBBLIGATORIO. Tipo dell'evento da notificare. DEVE essere una stringa case-sensitive e DEVE supportare i seguenti valori: • *credential_accepted*: quando l'Attestato Elettronico è stato memorizzato con successo nell'Istanza del Wallet. • *credential_deleted*: quando l'emissione non riuscita dell'Attestato Elettronico è stata causata da un'azione dell'utente. • *credential_failure*: in tutti gli altri casi di insuccesso.	Sezione 11.1 di [Open ID4VCI].
event_description	OPZIONALE. Testo ASCII *human-readable* [USASCII] che fornisce informazioni aggiuntive, da utilizzare per informare in merito all'evento verificatosi. I valori per il parametro event_description NON DEVONO includere caratteri al di fuori dell'insieme *%x20-21 / %x23-5B / %x5D-7E*.	Sezione 11.1 di [Open ID4VCI].

Notification Response

La Notification Response DEVE utilizzare un *status code HTTP 204 (No Content)*, come raccomandato nella Sezione 11.2 di [OpenID4VCI].

In caso di errori, si DEVE seguire quanto descritto nella Sezione 11.3 di [OpenID4VCI].

Nel caso in cui la Notification Request non contenga un Access Token valido, il Notification Endpoint restituisce una response di errore come definito nella Sezione 3 del [**RFC 6750**]. Se si verifica qualsiasi altro errore, il Credential Issuer DEVE restituire una response di errore come definito nella Sezione 11.3 di [OpenID4VCI]. La response DEVE utilizzare il content type *application/json* e DEVE includere i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma *human-readable* che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una response di errore.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
```

```
{
    "error": "invalid_notification_id",
    "error_description": "The notification_id field is not valid"
}
```

Nella seguente tabella sono elencati i *Status Code HTTP* e i relativi codici di errore supportati per la response di errore:

Codice di Stato	codice di errore	Descrizione
400 Bad Request [OBBLIGATORIO]	`invalid_notification_id`	Il Credential Issuer non può soddisfare la richiesta a causa del parametro `notification_id` non valido. Sezione 11.3 di [OpenID4VCI].
400 Bad Request [OBBLIGATORIO]	`invalid_notification_request`	Il Credential Issuer non può soddisfare la richiesta a causa di parametri mancanti, parametro non valido o richiesta malformata. Sezione 11.3 di [OpenID4VCI].
500 Internal Server Error [OBBLIGATORIO]	`server_error`	Il Credential Issuer ha riscontrato un problema interno.
503 Service Unavailable [OBBLIGATORIO]	`temporarily_unavailable`	Il Credential Issuer è temporaneamente non disponibile.
504 Gateway Timeout [OPZIONALE]	-	Il Credential Issuer non può soddisfare la richiesta entro l'intervallo di tempo definito.

Correzione dati usando credential_failure

Secondo OpenID4VCI Sezione 11, in tutti gli altri casi di insuccesso `event` DEVE usare `credential_failure` e parametri addizionali della Notification Request POSSONO essere definiti e usati. Il Credential Issuer DEVE ignorare i parametri non riconosciuti.

Per la correzione dati iniziata dall'Utente, l'Istanza del Wallet DOVREBBE inviare una Notification Request con `event=credential_deleted` includendo parametri aggiuntivi per segnalare il contesto della correzione dati. Al minimo:

- `event_description`: descrizione concisa e leggibile della discrepanza.
- `failure_reason` (OPZIONALE): breve codice machine-readable, ad es. `data_correction_requested`.
- `correction_details` (OPZIONALE): oggetto con campi minimi per indicare gli attributi impattati senza valori sensibili (es. soli identificativi degli attributi).

Nell'eventualità di aver ricevuto un rigetto motivato da parte dell'Utente, il Credential DOVREBBE inoltrare il messaggio dell'utente contenente quel motivo alla Fonte Autentica.

È RACCOMANDATO di utilizzare endpoint specifici PDND, forniti da Authentic Source, per facilitare questo scambio di dati.

13.2.3 Catalogo degli e-Service PDND del Credential Issuer

I Credential Issuer DEVONO fornire i seguenti e-service attraverso PDND per:

- fornire statistiche sugli Attestati Elettronici emessi

Get Statistics

Descrizione	Questo servizio restituisce dati statistici sugli Attestati Elettronici emessi.
Erogatore	Credential Issuer
Fruitore	Terza Parte Autorizzata

13.3 Endpoint della Relying Party

La Relying Party DEVE esporre una serie completa di endpoint per supportare l'ecosistema IT-Wallet completo, inclusi sia i flussi di presentazione remoti che di prossimità.

13.3.1 Endpoint per Flussi Remoti della Relying Party

La Relying Party DEVE esporre una serie di endpoint per supportare i flussi di presentazione remoti come definiti in OpenID4VP 1.0. Questi endpoint abilitano la verifica sicura delle credenziali, l'instaurazione della fiducia e l'autenticazione dell'utente per modelli di interazione cross-device e same-device.

Nota

I test relativi agli endpoint per flussi remoti della Relying Party sono definiti nella matrice di test per presentazione remota (test-plans-remote-presentation:Matrice di Test per il Verificatore di Credenziali in Remoto).

Endpoint di Federazione

La Relying Party DEVE fornire la propria Entity Configuration attraverso l'endpoint `/.well-known/openid-federation`, secondo la Sezione *Entity Configuration* (pagina 126). Questo endpoint abilita l'instaurazione della fiducia e la scoperta delle capacità della Relying Party.

I dettagli tecnici sono forniti nella Sezione *Entity Configuration Relying Party* (pagina 372).

Endpoint per Flussi Remoti OpenID4VP

I seguenti endpoint sono richiesti per i flussi di presentazione remota OpenID4VP 1.0 come descritto in *Flusso Remoto* (pagina 509). Questi endpoint supportano sia i flussi Same Device che Cross Device:

Endpoint Request URI

L'Endpoint Request URI è dove la Relying Party fornisce il Request Object firmato all'Istanza del Wallet. Questo endpoint supporta sia i metodi GET che POST come definito nella specifica OpenID4VP 1.0.

Per i requisiti di implementazione dettagliati, vedere *Richiesta all'Endpoint URI Request* (pagina 522) e *Risposta dell'Endpoint URI Request* (pagina 524).

Endpoint Response URI

L'Endpoint Response URI riceve l'Authorization Response dall'Istanza del Wallet contenente la Verifiable Presentation. Questo endpoint elabora la presentazione e convalida le credenziali.

Per i requisiti di implementazione dettagliati, vedere *Authorization Response* (pagina 529) e *Risposta della Relying Party* (pagina 533).

Endpoint Status (Opzionale)

L'Endpoint Status è un endpoint opzionale che consente all'user-agent di monitorare il progresso del flusso di presentazione. Questo endpoint è particolarmente utile per i flussi Same Device dove l'user-agent deve sapere quando l'Istanza del Wallet ha completato la presentazione.

Per i requisiti di implementazione dettagliati, vedere *Status Endpoint* (pagina 535) e *Errori dello Status Endpoint* (pagina 536).

Endpoint di Gestione Dati Utente

Il seguente endpoint supporta la gestione dei dati utente e i requisiti di conformità alla privacy per i flussi remoti:

Endpoint di Cancellazione della Relying Party

L'Endpoint di Cancellazione, che è descritto in *Metadati della Relying Party* (pagina 375), consente alle Istanze di Wallet di richiedere la cancellazione degli attributi presentati alla Relying Party. La Relying Party DEVE richiedere l'autenticazione dell'Utente prima di procedere con la cancellazione degli attributi.

Richiesta di Cancellazione

La Richiesta di Cancellazione DEVE essere una richiesta GET all'Endpoint di Cancellazione. L'Istanza di Wallet DEVE anche supportare un meccanismo di callback che consenta allo User-Agent di notificare lo stato della richiesta all'Istanza di Wallet (e quindi all'Utente) una volta che viene restituita la Risposta di Cancellazione.

Di seguito è riportato un esempio non normativo di una Richiesta di Cancellazione in cui l'URL di callback viene passato come parametro di query.

```
GET /erasure-endpoint?callback_url=https://wallet-instance/erasure_response HTTP/1.1
Host: relying-party.example.org
```

Risposta di Cancellazione

Se la cancellazione di tutti gli attributi associati all'Utente è avvenuta con successo, la Risposta di Cancellazione DEVE restituire un Status Code HTTP 204.

Se invece la procedura di cancellazione degli attributi fallisce per qualsiasi circostanza, la Relying Party DEVE restituire una risposta di errore con `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

La seguente tabella elenca gli Status Code HTTP e i relativi Error Codes che DEVONO essere supportati per la Error Response:

Codice di Stato	Codice di Errore	Descrizione
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri richiesti (ad esempio, parametri di intestazione o asserzione di integrità), o include parametri non validi e sconosciuti.
`401 Unauthorized`	`unauthorized`	La richiesta non può essere soddisfatta in quanto l'autenticazione da parte dell'Utente risulta fallita o non valida.
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint di Cancellazione ha riscontrato un problema interno. (**RFC 6749#section-4.1.2.1**).
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint di Cancellazione è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico). (**RFC 6749#section-4.1.2.1**).

Di seguito è riportato un esempio di Error Response dall'Endpoint di Cancellazione:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
 "error": "server_error",
 "error_description": "The request cannot be fulfilled due to an internal server error."
}
```

Alla ricezione di una Error Response, l'Istanza di Wallet che ha effettuato la Richiesta di Cancellazione DEVE informare l'Utente della condizione di errore in modo appropriato.

Considerazioni di Sicurezza

Tutti gli endpoint della Relying Party DEVONO implementare appropriate misure di sicurezza:

- **Solo HTTPS**: Tutti gli endpoint DEVONO essere accessibili solo tramite HTTPS
- **Protezione Endpoint Mix-up**: Gli URL degli endpoint DEVONO essere attestati da terze parti fidate attraverso la Trust Chain
- **Validazione Input**: Tutti gli endpoint DEVONO convalidare i parametri di input e rifiutare richieste malformate
- **Rate Limiting**: Gli endpoint DOVREBBERO implementare rate limiting per prevenire abusi
- **Audit Logging**: Tutte le interazioni degli endpoint DOVREBBERO essere registrate per il monitoraggio della sicurezza

Per i requisiti di sicurezza dettagliati, vedere *Flusso Remoto* (pagina 509) e i casi di test pertinenti in test-plans-remote-presentation:Matrice di Test per il Verificatore di Credenziali in Remoto.

Note di Implementazione

- I dettagli di implementazione specifici per la maggior parte degli endpoint sono lasciati alla discrezione della Relying Party
- Gli endpoint DEVONO essere conformi alla specifica OpenID4VP 1.0 per i flussi remoti
- Gli endpoint per flussi di prossimità DEVONO supportare la gestione del ciclo di vita delle App di Verifica
- Tutti gli endpoint DEVONO essere scopribili attraverso l'Entity Configuration della Relying Party
- Le risposte di errore DEVONO seguire i codici di stato HTTP standard e includere appropriate descrizioni degli errori

Per una guida di implementazione completa, fare riferimento alle sezioni degli endpoint individuali e alle matrici di test per i requisiti di validazione.

13.3.2 Endpoint del Backend del Provider di Relying Party

La Relying Party DEVE esporre una serie di endpoint per gestire il ciclo di vita delle App di Verifica che utilizzano un servizio di backend remoto fornito dal loro Backend del Provider di Relying Party. Questi endpoint supportano i flussi di presentazione in prossimità fornendo generazione di nonce, registrazione delle chiavi hardware, convalida dell'integrità e rilascio del Certificato di Accesso. I dettagli specifici della loro implementazione sono lasciati alla discrezione della Relying Party.

Nota

I test relativi agli endpoint della Relying Party sono definiti nella matrice di test per presentazione remota (test-plans-remote-presentation:Matrice di Test per il Verificatore di Credenziali in Remoto) e prossimità (test-plans-proximity-presentation:Matrice di Test per il Verificatore di Credenziali in Prossimità).

Endpoint di Federazione della Relying Party

La Relying Party DEVE fornire la propria Entity Configuration attraverso l'Endpoint `/.well-known/openid-federation`, secondo la Sezione *Entity Configuration* (pagina 126). I dettagli tecnici sono forniti nella Sezione *Entity Configuration Relying Party* (pagina 372).

Endpoint Nonce della Relying Party

Il Nonce Endpoint della Relying Party consente all'App di Verifica di richiedere un `nonce` crittografico dal Backend del Provider di Relying Party. Il `nonce`, un codice monouso e casuale, serve per garantire l'unicità e prevenire replay attacks.

Ulteriori dettagli sulla Richiesta e Risposta Nonce sono forniti rispettivamente nelle Sezioni *Richiesta di Nonce dell'Applicazione Mobile* (pagina 710) e *Richiesta di Nonce dell'Applicazione Mobile* (pagina 710).

Endpoint di Inizializzazione dell'App di Verifica della Relying Party

L'Endpoint di Inizializzazione dell'App di Verifica consente l'inizializzazione delle App di Verifica, consistente nella registrazione di una coppia di Cryptographic Hardware Keys a lunga durata, memorizzate in modo sicuro.

Ulteriori dettagli sulla Richiesta e Risposta di Inizializzazione dell'App di Verifica sono forniti rispettivamente nelle Sezioni *Richiesta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 712) e *Risposta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 713).

Endpoint di Associazione Chiavi della Relying Party

Il Key Binding Endpoint della Relying Party consente alle App di Verifica di associare la coppia di chiavi appena creata, che sarà associata a un Certificato di Accesso, all'App di Verifica, basandosi su una dimostrazione di possesso delle Cryptographic Hardware Keys generate durante la fase di *Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 705). Prima di completare il processo, il Backend del Provider di Relying Party deve anche verificare l'integrità dell'App di Verifica.

Richiesta di Associazione Chiavi della Relying Party

Ulteriori dettagli sulla Relying Party Key Binding Request sono forniti nella sezione *Richiesta di Emissione della Wallet Instance Attestation* (pagina 576).

Le uniche differenze sono le seguenti:

- L'intestazione `typ` del JWT della Integrity Request assume il valore `rp-kb+jwt`.
- Il valore della dichiarazione `hardware_signature` viene ottenuto basandosi esclusivamente sul valore di `client_data_hash`.

Risposta di Associazione Chiavi della Relying Party

In caso di richiesta riuscita, il Backend del Provider di Relying Party fornisce una HTTP Response con Status Code `204 No Content`.

Di seguito è riportato un esempio non normativo di una Risposta alla Richiesta di Associazione Chiavi.

```
HTTP/1.1 204 No content
```

Se si verifica un errore durante il processo, viene restituita una risposta di errore. La risposta utilizza `application/json` come `Content-Type` e include i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in formato leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato (*WP_035* (pagina 781)).

Di seguito è riportato un esempio non normativo di una Key Binding Error Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The provided challenge is invalid, expired, or already used."
}
```

La seguente tabella elenca i codici di stato HTTP e i relativi codici di errore supportati per la risposta di errore, salvo diversa indicazione (*WP_036–0339* (pagina 781) and *WP_150–155* (pagina 814)):

HTTP Status Code	Error Code	Descrizione
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri obbligatori (ad esempio parametri di intestazione o Integrity Assertion), oppure include parametri non validi o sconosciuti.
`403 Forbidden`	`invalid_request`	The Verifier App è stata revocata.
`403 Forbidden`	`integrity_check_error`	Il dispositivo non soddisfa i requisiti minimi di sicurezza del Fornitore della Relying Party.
`403 Forbidden`	`invalid_request`	La firma della Integrity Request non è valida oppure non corrisponde alla chiave pubblica (JWK) associata.
`403 Forbidden`	`invalid_request`	La validazione della Integrity Assertion non è riuscita; la Integrity Assertion è stata manomessa o firmata in modo non corretto.
`403 Forbidden`	`invalid_request`	Il `nonce` fornito non è valido, è scaduto oppure è già stato utilizzato.
`403 Forbidden`	`invalid_request`	La Proof of Possession (`hardware_signature`) non è valida.
`403 Forbidden`	`invalid_request`	Il parametro `iss` non corrisponde all'identificatore URL previsto dal Fornitore della Relying Party.
`404 Not Found`	`not_found`	L'istanza dell'applicazione di verifica (Verifier App) non è stata trovata.
`422 Unprocessable Content` [OPZIONALE]	`validation_error`	La richiesta non rispetta il formato richiesto.
`500 Internal Server Error`	`server_error`	Si è verificato un errore interno del server durante l'elaborazione della richiesta.
`503 Service Unavailable`	`temporarily_unavailable`	Il servizio non è disponibile. Si prega di riprovare più tardi.

Endpoint del Certificato di Accesso della Relying Party

L'Endpoint del Certificato di Accesso della Relying Party consente alle App di Verifica di ottenere un Certificato di Accesso.

Richiesta del Certificato di Accesso della Relying Party

La Richiesta del Certificato di Accesso utilizza il metodo HTTP POST con `Content-Type` impostato su `application/json`.

La richiesta include il seguente parametro nel body:

Parametro	Descrizione	Riferimento
csr	Il CSR generato dall'App di Verifica, codificato nel formato `base64url` come definito in **RFC 2511**.	

Di seguito è riportato un esempio non normativo di una Richiesta di Certificato di Accesso.

```
POST /access-certificate HTTP/1.1
Host: relying-party.example.org
Content-Type: application/json

{
  "csr": "MIIBvzCCAa..."
}
```

Risposta del Certificato di Accesso della Relying Party

In caso di richiesta riuscita, l'Access Certificate Endpoint della Relying Party fornisce un HTTP Response con Status Code `200 OK` e il Certificato di Accesso. La Risposta dell'Access Certificate Endpoint, che utilizza `application/json` come `Content-Type`, include i seguenti parametri nel body:

Parametro	Descrizione	Riferimento
access_certificate	Il Certificato di Accesso generato dal CSR.	Questa specifica.

Di seguito è riportato un esempio non normativo di Risposta dall'Access Certificate Endpoint.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_certificate": "hajdnhaghSDGns..."
}
```

Se si verificano errori, l'Access Certificate Endpoint della Relying Party restituisce una Error Response. La risposta utilizza `application/json` come tipo di contenuto e include i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma leggibile che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una Error Response dell'Access Certificate Endpoint.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
    "error": "invalid_request",
    "error_description": "The public key in the CSR is different from the one associated with t
he Cryptographic Hardware Keys."
}
```

La seguente tabella elenca gli HTTP Status Code e i relativi codici di errore che DEVONO essere supportati per l'Error Response, se non diversamente specificato:

Codice di Stato HTTP	Codice di Errore	Descrizione
`400 Bad Request`	`bad_request`	La richiesta non è conforme allo standard, mancano parametri richiesti (ad esempio, `integrity_assertion`), oppure sono includi parametri non validi e sconosciuti.
`403 Forbidden`	`invalid_request`	La chiave pubblica nel CSR non corrisponde alla chiave pubblica associata alle Cryptographic Hardware Keys.
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint ha riscontrato un problema interno.
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

Endpoint di Cancellazione della Relying Party del Provider Backend

L'Endpoint di Cancellazione consente alle Istanze di Wallet di richiedere la cancellazione degli attributi presentati alla Relying Party, supportando i diritti di privacy dell'utente e la conformità normativa.

Per i requisiti di implementazione dettagliati, vedere relying-party-remote-flow-endpoints: Endpoint di Cancellazione della Relying Party.

13.4 Endpoint delle Fonti Autentiche

13.4.1 Catalogo degli e-Service PDND delle Fonti Autentiche

Le Fonti Autentiche pubbliche DEVONO realizzare e rendere disponibile tramite PDND il seguente e-service al fine di rilasciare al Fornitore di Attestati Elettronici gli Attributi dell'Utente necessari per l'emissione di un Attestato Elettronico.

L'e-service è descritto tramite una specifica OpenAPI in cui sono dettagliati i messaggi di richiesta, risposta ed errore.

Nota

La Specifica OpenAPI è disponibile *Specifica OpenAPI della Fonte Autentica PDND* (pagina 753). Questa specifica OpenAPI può essere estesa dalle Fonti Autentiche, infatti, l'array `attributeClaims` PUÒ contenere proprietà aggiuntive specifiche di una particolare Credenziale. Queste proprietà aggiuntive, così come definito nella specifica OpenAPI, saranno inserite nella Credenziale dal Credential Issuer.

Get Attribute Claims

Descrizione	Questo servizio fornisce al Fornitore di Attestati Elettronici tutti gli attributi dell'Utente necessari per il rilascio di un Attestato Elettronico.
Erogatore	Fonte Autentica
Fruitore	Fornitore di Attestato Elettronico

Questo e-Service DEVE essere invocato dal Credential Issuer nei seguenti casi:

- Durante il flusso di emissione delle Credenziali per recuperare i relativi dataset. In questo caso, il claim `object_id` NON DEVE essere utilizzato nel payload della request dell'e-Service e la response dell'e-Service DEVE includere tutti i dataset delle credenziali in stato `VALID` e non scaduti amministrativamente (ovvero, `expiry_date` > data corrente della request dell'e-service).
- Dopo aver ricevuto una notifica di aggiornamento tramite Signal Hub. In questo caso, il claim `object_id` DEVE essere incluso nel payload della request dell'e-service e la risposta dell'e-Service DEVE restituire solamente il dataset delle credenziali identificato dal claim `object_id`, indipendentemente dal suo stato.

Nota

La Fonte Autentica e il Credential Issuer DEVONO implementare la logica necessaria per tenere traccia delle richieste e delle risposte scambiate tramite questo e-Service, al fine di essere in grado di correlarle con la relativa emissione di un Attestato Elettronico. In particolare,

- entrambi DEVONO salvare il valore `object_id` contenuto nel payload della risposta per gestire i Segnali relativi alla disponibilità degli Attributi utili all'emissione in *deferred* di un Attestato Elettronico (vedere *Elaborazione dei Segnali* (pagina 644)). Qualsiasi notifica successiva relativa a un set di dati specifico DEVE essere gestita tramite `object_id`.
- la Fonte Autentica DEVE registrare il valore datetime fornito all'interno del parametro `last_updated`, che indica data e orario dell'ultima volta che gli Attributi dell'Utente sono stati aggiornati nel database della Fonte Autentica;
- il Credential Issuer DEVE leggere il valore `last_updated` ricevuto nella risposta per essere in grado di verificare se gli Attributi dell'Utente sono cambiati dall'ultima emissione di un Attestato Elettronico.

Mapping degli Stati del Ciclo di Vita degli Attestati Elettronici

Per garantire la coerenza tra il "Ciclo di Vita degli Attestati Elettronici" documentato in *Ciclo di Vita degli Attestati Elettronici* (pagina 422) e l'Enum status delle OpenAPI, la seguente mappatura e logica operativa DEVE essere applicata per il campo `status` negli `attributeClaims`.

Direzionalità e Responsabilità: I cambiamenti di stato di un Attestato Elettronico a livello di Fornitore di Attestati Elettronici NON implicano un cambiamento presso la Fonte Autentica. Al contrario, qualsiasi cambiamento di stato di un dataset presso la Fonte Autentica DEVE essere elaborato dal Fornitore di Attestati Elettronici per aggiornare lo stato tecnico dell'Attestato Elettronico.

Guida Operativa:

- **Validità Tecnica vs Amministrativa**: Gli Attestati Elettronici distinguono tra una **validità tecnica** stabilita dal Fornitore di Attestati Elettronici (claim `iat` ed `exp`) e una **validità amministrativa** determinata dalla Fonte Autentica (claim `issuance_date` ed `expiry_date`).
- **Gerarchia delle Scadenze**: La scadenza tecnica (`exp`) NON DEVE essere successiva alla scadenza amministrativa (`expiry_date`). Per esempio, una patente di guida può essere amministrativamente valida per 10 anni, mentre l'Attestato Elettronico emesso può avere una scadenza tecnica di 1 anno.

- **Riemissione**: Se un Attestato Elettronico raggiunge la sua scadenza tecnica (`exp`) ma il dataset è ancora amministrativamente valido, lo stato OpenAPI rimane `VALID`, consentendo all'Attestato Elettronico di essere riemesso più volte entro l'arco temporale amministrativo.
- **Verifica dei Metadati**: Il Fornitore di Attestati Elettronici DEVE verificare l'effettiva usabilità controllando sia i claim tecnici che le date amministrative.
- **Irreversibilità**: Dopo una transizione a `INVALID`, l'Attestato Elettronico non può tornare a uno stato `VALID`. È richiesta una nuova emissione per un nuovo dataset. Questo si applica sia alla revoca esplicita che alla scadenza amministrativa.
- **Elaborazione dei Segnali**: I Segnali DEVONO essere elaborati sequenzialmente. Se un Segnale invalida un Attestato Elettronico, i successivi Segnali di correzione per lo stesso oggetto vengono ignorati.

Mapping degli Stati e Logica dei Casi:

Il Fornitore di Attestati Elettronici DEVE aggiornare lo `status` dell'Attestato Elettronico in base ai Segnali ricevuti dalla Fonte Autentica via Signal Hub (`signalType=UPDATE`):

- **Revoca**: Se un dataset viene revocato presso la Fonte Autentica (stato `INVALID`), il Fornitore di Attestati Elettronici DEVE revocare gli Attestati Elettronici che utilizzano quel dataset (transizione di stato da `VALID/SUSPENDED` a `INVALID`).
- **Sospensione**: Se un dataset viene sospeso presso la Fonte Autentica (stato `SUSPENDED`), il Fornitore di Attestati Elettronici DEVE sospendere gli Attestati Elettronici (transizione di stato da `VALID` a `SUSPENDED`).
- **Ripristino**: Se un dataset sospeso torna a essere `VALID` presso la Fonte Autentica, il Fornitore di Attestati Elettronici DEVE ripristinare la validità dell'Attestato Elettronico (transizione di stato da `SUSPENDED` a `VALID`).
- **Modifica**: Se un dataset viene modificato ma rimane `VALID` presso la Fonte Autentica, il Fornitore di Attestati Elettronici — rilevando il cambiamento tramite il campo `last_updated` — assegna lo stato tecnico `ATTRIBUTE_UPDATE` all'Attestato Elettronico. Questo avvia un flusso di riemissione quando l'Istanza del Wallet controlla lo stato.
- **Scadenza Amministrativa:**
  - **Scenario A (Basato sui metadati)**: Se `expiry_date` è stata condivisa con il Fornitore di Attestati Elettronici nei metadati, la Fonte Autentica NON invia Segnali alla scadenza; il Fornitore di Attestati Elettronici gestisce il ciclo di vita in modo indipendente garantendo che l'`exp` tecnico sia <= `expiry_date`.
  - **Scenario B (Basato sui segnali)**: Se `expiry_date` NON è presente nei metadati e il dataset scade, la Fonte Autentica DEVE impostare il suo stato su `INVALID` e inviare un Segnale via Signal Hub; il Fornitore di Attestati Elettronici revoca quindi gli Attestati Elettronici (transizione di stato a `INVALID`).

Esempio di risposta della Authentic Source

La risposta DEVE avere come HTTP Content-Type `application/json`. Di seguito un esempio concreto con dati fittizi per chiarire forma e contenuto attesi.


```
{
  "userClaims": {
    "given_name": "Mario",
    "family_name": "Rossi",
    "birth_date": "1980-01-10",
    "birth_place": "Roma",
    "tax_id_code": "TINIT-RSSMRA80A01H501Z",
    "personal_administrative_number": "12345A123A"
  },
  "attributeClaims": [
    {
      "object_id": "6F9619FF-8B86-D011-B42D-00C04FC964FF",
      "institute_name": "Nome Istituto Universitario",
      "programme_type_name": "Laurea Magistrale",
      "degree_course_name": "Computer Science - Informatica",
      "academic_qualification_date": "2025-06-25",
      "issuance_date": "2025-06-25",
      "...": ""
    },
    {
      "object_id": "7A0720AB-9C97-E122-C53E-11D05FD075GG",
      "institute_name": "Nome Istituto Universitario",
      "programme_type_name": "Laurea Triennale",
      "degree_course_name": "Informatica",
      "academic_qualification_date": "2022-11-27",
      "issuance_date": "2022-11-27",
      "...": ""
    }
  ],
  "metadataClaims": [
    {
      "object_id": "6F9619FF-8B86-D011-B42D-00C04FC964FF",
      "description": "Laurea Magistrale in Informatica 2025",
      "status": "VALID",
      "last_updated": "2025-09-15T10:30:00Z"
    },
    {
      "object_id": "7A0720AB-9C97-E122-C53E-11D05FD075GG",
      "description": "Laurea Triennale in Informatica 2027",
      "status": "VALID",
      "last_updated": "2023-01-10T08:00:00Z"
    }
```

```
  ]
}
```

In sintesi:

- **userClaims**: dati anagrafici dell'utente (nome, cognome, data/luogo di nascita, codice fiscale o numero di identificazione). Almeno uno tra `tax_id_code` e `personal_administrative_number` è richiesto se si forniscono user claims.
- **attributeClaims**: array di dataset; ogni elemento **DEVE** contenere `object_id`, `status` (VALID | INVALID | SUSPENDED), `last_updated` (formato ISO 8601), più eventuali attributi aggiuntivi specifici del dataset (es. `nationality`, `residence_address`).
- **metadataClaims**: array di metadati per dataset (`object_id` obbligatorio; `issuance_date` e `expiry_date` opzionali).
- **interval**: obbligatorio se non è presente il parametro `claims` nella richiesta; indica i secondi da attendere prima di ripetere la richiesta (es. 864000 = 10 giorni).

La risposta in caso di successo (HTTP 200) restituisce un oggetto `CredentialClaimsResponse` formattato come **Payload JSON**, accompagnato dagli header di integrità `Agid-JWT-Signature` e `Digest`.

Verifica della Firma e Gestione Chiavi

Il Credential Issuer (Fruitore) DEVE verificare l'integrità e l'autenticità della risposta JSON validando gli header `Agid-JWT-Signature` e `Digest` ricevuti dalla Fonte Autentica.

Il processo di verifica e recupero delle chiavi DEVE seguire rigorosamente il pattern di sicurezza **INTEGRITY_REST_02** definito per gli **e-Service PDND**. Si rimanda all'Appendice tecnica (Sezione *e-Service PDND* (pagina 714)) per i dettagli sulla validazione della firma JWT e per le specifiche sul recupero della chiave pubblica dell'Erogatore tramite API di Interoperabilità.

Avvertimento

Non sono ammessi meccanismi alternativi di distribuzione del materiale crittografico (es. endpoint `.well-known` pubblici esposti direttamente dalla Fonte Autentica o distribuzione *out-of-band*). La gestione del trust DEVE rimanere centralizzata all'interno del perimetro dell'infrastruttura PDND come descritto nei riferimenti sopra citati.

13.5 Endpoint di PDND Signal Hub

All'interno della piattaforma PDND, Signal Hub funge da intermediario tra un Provider PDND e i suoi Consumer PDND per facilitare le segnalazioni di variazioni dei dati. Per abilitare tale funzionalità, il gestore della piattaforma PDND, di seguito denominato PDND Manager, agendo come e-Service PDNDProvider, fornisce due e-Service PDND di Signal Hub:

- l'e-Service di Raccolta Segnali che viene utilizzato dai e-Service PDNDProvider per depositare i Segnali; in questo caso, il e-Service PDNDProvider agisce come Consumer dell'e-Service di Raccolta Segnali;
- l'e-Service di Distribuzione Segnali che viene utilizzato dai e-Service PDNDConsumer per recuperare i Segnali raccolti; in questo caso, il e-Service PDNDConsumer agisce anche come Consumer dell'e-Service di Distribuzione Segnali.

Al fine di proteggere la privacy del soggetto del Segnale, il PDND Manager richiede a ciascun e-Service PDNDProvider di pseudonimizzare l'identificativo del soggetto utilizzato all'interno dei Segnali e di configurare un endpoint di pseudonimizzazione per il proprio PDND e-Service. Questo endpoint di pseudonimizzazione viene utilizzato dagli e-Service Consumer per ottenere l'algoritmo di pseudonimizzazione al fine di calcolare lo Pseudonimo del soggetto del Segnale. Solo il e-Service PDNDProvider e i suoi e-Service PDNDConsumer sono in grado di collegare un Segnale ai dati personali del soggetto, mentre il PDND Manager gestisce solo gli identificativi pseudonimizzati.

Per specifiche tecniche dettagliate e linee guida per l'implementazione, si prega di fare riferimento alla Signal Hub Guide.

Nel contesto dell'IT Wallet, le Fonti Autentiche interagiscono con Signal Hub per notificare ai Fornitori di Attestati Elettronici i cambiamenti nello stato e/o nel valore degli Attributi associati agli Attestati Elettronici. Nello specifico,

- la Fonte Autentica depositerà i Segnali in Signal Hub, svolgendo quindi il ruolo di PDND Consumer dell'e-Service di Raccolta Segnali;
- il Fornitore di Attestati Elettronici recupererà i Segnali da Signal Hub, e svolgerà quindi il ruolo di e-Service PDNDConsumer dell'e-Service di Distribuzione Segnali.

Nota

Nel contesto dell'IT Wallet, a causa della particolare natura dei dati scambiati, la pseudonimizzazione del soggetto del Segnale non è necessaria poiché l'identificativo è già opaco e non correlato al soggetto dell'Attestato Elettronico. Pertanto, la Fonte Autentica non ha bisogno di configurare un endpoint di pseudonimizzazione per i suoi e-Service.

Le Fonti Autentiche che utilizzano PDND DEVONO utilizzare gli e-Service di Signal Hub.

Di seguito viene fornita la descrizione di come le Fonti Autentiche e i Fornitori di Attestati Elettronici interagiscono con gli e-Service di Signal Hub insieme ai formati dettagliati delle request e delle response.

13.5.1 Prerequisiti

Prima di utilizzare Signal Hub, tutte le Fonti Autentiche DEVONO:

- essersi registrate come Provider del loro e-Service presso PDND;
- essersi registrate come Consumer dell'e-Service di Raccolta Segnali di Signal Hub;

Prima di utilizzare Signal Hub, tutti i Fornitori di Attestati Elettronici DEVONO:

- essersi registrati come Consumer degli e-Service delle Fonti Autentiche pertinenti;
- essersi registrati come Consumer dell'e-Service di Distribuzione Segnali di Signal Hub;

13.5.2 e-Service di Signal Hub

Questa sezione descrive gli endpoint disponibili per gli e-Service di Raccolta e Distribuzione di Signal Hub, incluse le loro funzionalità e i formati di richiesta e risposta previsti.

e-Service di Raccolta Segnali

Le Fonti Autentiche nell'ecosistema IT Wallet utilizzano l'e-Service di Raccolta Segnali per:

- notificare al Fornitore di Attestati Elettronici un cambiamento di stato e/o valore di un Attributo specifico associato a un Attestato Elettronico emesso dal Fornitore di Attestati Elettronici;
- notificare al Fornitore di Attestati Elettronici la disponibilità degli Attributi relativi a un Attestato Elettronico specifico che un Utente ha richiesto nel suo Wallet.

L'ultimo caso, denominato *deferred issuance*, si verifica quando il Fornitore di Attestati Elettronici ha richiesto gli Attributi di un Attestato Elettronico dalla Fonte Autentica (invocando l'endpoint PDND *Get Attribute Claims* (pagina 634)) e la Fonte Autentica non può rispondere immediatamente con gli Attributi richiesti. Pertanto, la Fonte Autentica notifica al Fornitore di Attestati Elettronici tramite Signal Hub in un momento successivo che gli Attributi richiesti sono ora disponibili.

L'endpoint dell'e-Service di Raccolta Segnali viene utilizzato dalle Fonti Autentiche per depositare i Segnali in Signal Hub tramite una request di Raccolta Segnali. Quest'ultima DEVE essere una richiesta POST con `Content-Type` impostato su `application/json`, il cui header DEVE

avere i parametri descritti nel Signal Hub Guide e il cui body DEVE contenere i seguenti parametri:

Nome Parametro	Descrizione
signalId	OBBLIGATORIO. Numero intero positivo a 64 bit identificativo del Segnale.
objectType	OBBLIGATORIO. Questo è un campo libero che la Fonte Autentica PUÒ utilizzare per specificare ulteriormente il Segnale.
objectId	OBBLIGATORIO. Il soggetto a cui è legato il Segnale. DEVE essere impostato sul valore *Object_id* (`object_id`) che la Fonte Autentica ha utilizzato nel payload della risposta del *get attributes* verso il Credential Issuer (vedi *Get Attribute Claims* (pagina 634)). Il `signalType` del Segnale DEVE essere valorizzato con uno dei seguenti: • `CREATE`, al fine di notificare la disponibilità degli attributi relativi ad un specifico Attestato Elettronico. • `UPDATE`, al fine di notificare l'aggiornamento degli attributi relativi ad un specifico Attestato Elettronico.
signalType	OBBLIGATORIO. Tipo di Segnale. DEVE essere uno dei seguenti: • `UPDATE`, quando il Segnale si riferisce a un cambiamento nello stato e/o valore di un Attributo specifico associato a un Attestato Elettronico; • `CREATE`, quando il Segnale si riferisce alla disponibilità degli Attributi di un Attestato Elettronico specifico;
eserviceId	OBBLIGATORIO. e-Service a cui è legato il Segnale. DEVE corrispondere al valore dell'Id dell'e-Service di cui la Fonte Autentica è Provider.

Nota

Nel Deferred Issuance Flow, cioè quando la Fonte Autentica notifica al Fornitore di Attestati Elettronici la disponibilità degli Attributi dell'Attestato Elettronico tramite Signal Hub; entrambe le entità DEVONO tenere traccia del valore `object_id` della Fonte Autentica utilizzato nella risposta del *get attributes*. Questo è necessario poiché l'`objectId` del Segnale DEVE essere impostato su quel valore `object_id` quando il Segnale ha il `signalType` valorizzato con `CREATE`.

Un esempio non normativo di richiesta di richiesta di Raccolta Segnali può essere trovato nel Signal Hub push.

La response dell'e-Service di Raccolta Segnale è specificata nel Signal Hub push e ha `Content-Type` impostato su `application/json`. Il payload contiene il parametro del body:

Nome Parametro	Descrizione
signalId	OBBLIGATORIO. L'identificativo del Segnale che è stato raccolto con successo dall'e-Service di Raccolta Segnali.

Se si verifica un errore, la response DEVE aderire alla specifica definita nel Signal Hub push-yaml.

Nota

La Specifica OpenAPI completa dell'e-Service di Raccolta Segnali è disponibile nel Signal Hub push-yaml.

La Fonte Autentica DEVE implementare la logica necessaria per gestire le richieste all'endpoint dell'e-Service di Raccolta Segnali, nel fare ciò deve considerare i seguenti aspetti:

- I Segnali vengono inviati per e-Service PDND, pertanto la Fonte Autentica DOVREBBE implementare un ciclo di deposito Segnali per ogni e-Service ID di cui è Provider PDND;
- I Segnali sono etichettati da un identificativo univoco, il `signalId`, che è un numero intero positivo a 64 bit. Il `signalId` DEVE essere incrementato di 1 per ogni nuovo Segnale che la Fonte Autentica desidera depositare nell'endpoint dell'e-Service di Raccolta Segnali. Spetta alla Fonte Autentica tenere traccia dell'ultimo `signalId` che ha inviato. I Segnali con valori `signalId` più bassi sono considerati più vecchi dall'endpoint dell'e-Service di Raccolta Segnali e genereranno un errore quando ricevuti.

e-Service di Distribuzione Segnali

I Fornitori di Attestati Elettronici nell'ecosistema IT Wallet utilizzano l'e-Service di Distribuzione Segnali per:

- verificare i cambiamenti nello stato e/o il valore di un Attributo specifico associato a un Attestato Elettronico emesso dal Fornitore di Attestati Elettronici stesso;
- verificare la disponibilità degli Attributi relativi a un Attestato Elettronico richiesto da un Utente.

L'endpoint dell'e-Service di Distribuzione Segnali viene utilizzato dai Fornitori di Attestati Elettronici per recuperare i Segnali da Signal Hub tramite una richiesta di Distribuzione Segnali. Quest'ultima DEVE essere una richiesta HTTP con metodo GET e DEVE avere i seguenti parametri:

- Parametri di Path:

  - `eserviceId`. OBBLIGATORIO. e-Service a cui è legato il Segnale. DEVE corrispondere al valore dell'Id dell'e-Service di cui il Fornitore di Attestati Elettronici è Consumer.

- Parametri di Query:
  - `signalId`. OPZIONALE. Intero che rappresenta l'ultimo numero di Segnale elaborato dal Fornitore di Attestati Elettronici. L'e-Service di Distribuzione Segnali risponderà con Segnali aventi valori `signalId` progressivamente maggiori. Se non specificato, il valore predefinito è il valore `signalId` più basso disponibile nell'e-Service di Distribuzione Segnali.
  - `size`. OPZIONALE. Intero che rappresenta il numero massimo di Segnali da restituire nella risposta di Distribuzione Segnali. Se non specificato, il valore predefinito è `10`.
- Parametri degli Header: questi sono quelli descritti in Signal Hub pull.

Se la richiesta di Distribuzione Segnali viene elaborata correttamente, l'e-Service risponderà con Status Code:

- HTTP 200 OK se la richiesta è formata correttamente e non ci sono più Segnali da richiedere;
- HTTP 206 OK se la richiesta è formata correttamente ma ci sono più Segnali da richiedere.

Indipendentemente dal codice di risposta utilizzato, la risposta ha `Content-Type` impostato su `application/json` e i parametri dell'header indicati nel Signal Hub pull. Il parametro del body `lastSignalId`, che fa riferimento al `signalId` dell'ultimo Segnale trasmesso dall'e-Service di Distribuzione Segnali, viene aggiunto al payload della risposta.

In Signal Hub pull si possono trovare esempi non normativi di richieste e risposte di Distribuzione Segnali.

Se si verifica un errore durante l'analisi della request, la risposta DEVE aderire al formato di errore definito in Signal Hub pull-yaml.

Nota

La Specifica OpenAPI completa dell'e-Service di Raccolta Segnali è disponibile in Signal Hub pull-yaml.

Il Fornitore di Attestati Elettronici DEVE implementare la logica necessaria per gestire il polling presso l'endpoint dell'e-Service di Distribuzione Segnali, nel fare ciò deve considerare i seguenti aspetti:

- I Segnali vengono richiesti e recuperati per e-Service PDND, il che significa che il Fornitore di Attestati Elettronici DEVE implementare un ciclo di polling per ogni e-Service;

- il periodo di conservazione dei Segnali in Signal Hub è specificato nella Signal Hub Guide. I Segnali più vecchi del periodo di conservazione specificato non saranno disponibili per il recupero;
- Signal Hub non tiene traccia dell'ultimo `signalId` notificato ai Fornitori di Attestati Elettronici. Ogni Fornitore di Attestati Elettronici DEVE tenere traccia dell'ultimo `signalId` che ha ricevuto per ogni e-Service ID PDND;
- l'endpoint dell'e-Service di Distribuzione Segnali restituisce i Segnali in batch. La dimensione massima del batch è specificata nella Signal Hub Guide. L'e-Service di Distribuzione Segnali indicherà se sono disponibili Segnali aggiuntivi per il recupero.

13.5.3 Elaborazione dei Segnali

Dopo che i Segnali sono stati recuperati con successo dal Fornitore di Attestati Elettronici, quest'ultimo DEVE elaborarli come segue:

- Per ogni Segnale, il Fornitore di Attestati Elettronici DEVE verificare il valore `SignalType`:
  - se il `SignalType` del Segnale è `UPDATE` (l'`objectId` fa riferimento al *Object_id* della Fonte Autentica), lo stato e/o il valore dell'Attributo associato a un Attestato Elettronico necessitano di aggiornamenti;
  - se il `SignalType` del Segnale è `CREATE` (l'`objectId` corrisponde al *Object_id* della Fonte Autentica), gli Attributi richiesti di un Attestato Elettronico specifico sono ora disponibili;

  Se l'`objectId` non corrisponde ad alcun identificativo valido noto al Fornitore di Attestati Elettronici, il Segnale DEVE essere ignorato. Altrimenti, se corrisponde a un identificativo noto e valido, il Fornitore di Attestati Elettronici DEVE utilizzare l'endpoint PDND *Get Attribute Claims* (pagina 634) della Fonte Autentica per recuperare le informazioni aggiornate e, se possibile, applicare il nuovo stato/attributo all'Attestato Elettronico corrispondente.

  Quando il Segnale è stato elaborato, il Fornitore di Attestati Elettronici passerà al Segnale successivo e aggiornerà il suo contatore `signalId`; oppure, se non ci sono più Segnali da elaborare, riprenderà il polling.

Avvertimento

Dati i pattern di sicurezza attualmente supportati da Signal Hub, se la Fonte Autentica richiede il pattern di sicurezza *AUDIT_REST_02* dal Fornitore di Attestati Elettronici, quest'ultimo DEVE revocare l'Attestato Elettronico referenziato nei Segnali con `signalType` `UPDATE` non potendo contattare la Fonte Autentica per recuperare le informazioni aggiornate

senza aver prima autenticato l'Utente. **In questo scenario, la revoca è l'unica azione ammessa.**

CAPITOLO 14

Algoritmi Crittografici

Questa sezione elenca gli algoritmi crittografici utilizzati nell'ecosistema IT Wallet. Gli identificatori degli algoritmi sono espressi principalmente come valori JOSE `alg`; quando viene utilizzato COSE, i corrispondenti algoritmi COSE sono indicati esplicitamente (ad esempio, ESP256 rispetto a ES256).

I seguenti algoritmi DEVONO essere supportati:

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
ES256	Elliptic Curve Digital Signature Algorithm (ECDSA) utilizzando P-256 e SHA-256.	Firma	**RFC 7518**.
ES384	ECDSA utilizzando P-384 e SHA-384.	Firma	**RFC 7518**.
ES512	ECDSA utilizzando P-521 e SHA-512.	Firma	**RFC 7518**.
ESP256 (COSE, `-9`)	ECDSA utilizzando P-256 e SHA-256.	Firma	**RFC 9864**.
ESP384 (COSE, `-51`)	ECDSA utilizzando P-384 e SHA-384.	Firma	**RFC 9864**.
ESP512 (COSE, `-52`)	ECDSA utilizzando P-521 e SHA-512.	Firma	**RFC 9864**.

continues on next page

Tabella 14.1 – continua dalla pagina precedente

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
RSA-OAEP-256	RSA Encryption Scheme con Optimal Asymmetric Encryption Padding (OAEP) utilizzando la funzione di hash SHA-256 e la funzione di generazione MGF1 con SHA-256.	Cifratura delle Chiavi	**RFC 7516**, **RFC 7518**.
ECDH-ES	Elliptic Curve Diffie-Hellman (ECDH) Ephemeral-Static key agreement.	Accordo di chiave	**RFC 7516**, **RFC 7518**.
A128CBC-HS256	Cifratura AES in modalità Cipher Block Chaining con valore Initial Vector a 128 bit, più autenticazione HMAC utilizzando SHA-256 e troncando HMAC a 128 bit.	Cifratura del Contenuto	**RFC 7516**, **RFC 7518**.
A256CBC-HS512	Cifratura AES in modalità Cipher Block Chaining con valore Initial Vector a 256 bit, più autenticazione HMAC utilizzando SHA-512 e troncando HMAC a 256 bit.	Cifratura del Contenuto	**RFC 7516**, **RFC 7518**.
A128GCM	Cifratura AES con Galois/Counter Mode e una lunghezza della chiave di 128.	Cifratura del Contenuto	**RFC 7516**, **RFC 7518**.
A256GCM	Cifratura AES con Galois/Counter Mode e una lunghezza della chiave di 256.	Cifratura del Contenuto	**RFC 7516**, **RFC 7518**.

Per le Credenziali emesse in formato mdoc, i seguenti algoritmi DEVONO essere supportati:

Algoritmo	Descrizione	Operazioni	Riferimenti
ECKA-DH	Elliptic Curve Key Agreement Algorithm - Diffie-Hellman.	Accordo di chiave	BSI TR-03111.
HKDF	HMAC-based Key Derivation Function.	Derivazione della chiave di sessione	**RFC 5869**.
AES-256-GCM	Advanced Encryption Standard con Galois/ Counter Mode e una lunghezza della chiave di 256.	Cifratura della sessione	NIST SP 800-38D.

Si RACCOMANDA di supportare i seguenti algoritmi:

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
PS256	RSASSA (RSA with Signature Scheme Appendix) con padding PSS (Probabilistic Signature Scheme) utilizzando la funzione di hash SHA-256 e la funzione di generazione della maschera MGF1 con SHA-256.	Firma	**RFC 7518**.
PS384	RSASSA con padding PSS utilizzando la funzione di hash SHA-384 e la funzione di generazione della maschera MGF1 con SHA-384.	Firma	**RFC 7518**.
PS512	RSASSA con padding PSS utilizzando la funzione di hash SHA-512 e la funzione di generazione della maschera MGF1 con SHA-512.	Firma	**RFC 7518**.

continues on next page

Tabella 14.3 – continua dalla pagina precedente

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
ECDH-ES	Elliptic Curve Diffie-Hellman (ECDH) Ephemeral Static key agreement utilizzando Concat Key Derivation Function (KDF).	Cifratura delle Chiavi	**RFC 7518**.
ECDH-ES+ A128KW	ECDH-ES utilizzando Concat KDF e content encryption key (CEK) avvolta utilizzando AES con una lunghezza della chiave di 128 (A128KW).	Cifratura delle Chiavi	**RFC 7518**.
ECDH-ES+ A256KW	ECDH-ES utilizzando Concat KDF e content encryption key (CEK) avvolta utilizzando AES con una lunghezza della chiave di 256 (A256KW).	Cifratura delle Chiavi	**RFC 7518**.

I seguenti algoritmi NON DEVONO essere supportati:

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
none	•	Firma	**RFC 7518**.
RSA_1_5	RSAES con schema di padding PKCS1-v1_5. L'uso di questo algoritmo generalmente non è raccomandato.	Cifratura delle Chiavi	**RFC 7516**, [Security Vulnerability], [SOG-IS].
RSA-OAEP	RSA Encryption Scheme con Optimal Asymmetric Encryption Padding (OAEP) utilizzando parametri predefiniti.	Cifratura delle Chiavi	**RFC 7518**, [SOG-IS].

continues on next page

Tabella 14.4 – continua dalla pagina precedente

Valore del parametro 'alg' dell'algoritmo	Descrizione	Operazioni	Riferimenti
HS256	HMAC utilizzando SHA-256.	Firma	**RFC 7518**.
HS384	HMAC utilizzando SHA-384.	Firma	**RFC 7518**.
HS512	HMAC utilizzando SHA-512	Firma	**RFC 7518**.

CAPITOLO 15

Considerazioni di Sicurezza e Privacy

Questa sezione fornisce un'analisi informale della sicurezza della specifica IT-Wallet analizzando la conformità con i requisiti di sicurezza e privacy identificati in [OpenID4VC-SecTrust].

Nota

Poiché [OpenID4VC-SecTrust] è ancora in fase di sviluppo, le considerazioni di sicurezza e privacy descritte potrebbero cambiare in futuro.

Nota

Il focus dell'analisi è la conformità delle scelte progettuali nella specifica IT-Wallet rispetto ai protocolli OpenID4VC. Sono attualmente fuori ambito *(i)* l'analisi della progettazione del flusso di prossimità basato su ISO 18013-5, e *(ii)* l'analisi dell'implementazione; di conseguenza 7 requisiti specificamente relativi all'implementazione non sono considerati (ad esempio, SV-00: Il Verificatore deve implementare il protocollo in modo sicuro e corretto).

Come in [OpenID4VC-SecTrust], tutti i requisiti sono numerati per riferimento. Insieme al rispettivo componente che deve implementare il requisito:

- **CF**: Formato della Credenziale;
- **P**: Protocollo;
- **E**: Ecosistema;

- **I**: Fornitore di Credenziali;
- **V**: Verificatore di Attestati Elettronici;
- **W**: Wallet.

Questa specifica aggiunge la categoria del requisito come prefisso:

- **SR**: Requisiti di Sicurezza;
- **PR**: Requisiti di Privacy;
- **SPR**: Requisiti di Sicurezza e Privacy.

L'identificatore finale utilizza il seguente spazio dei nomi: *categoria_requisito-componente-id*.

Per ogni requisito definito di seguito, questa sezione utilizza la descrizione definita in [Open ID4VC-SecTrust], specificando se il requisito è soddisfatto (completamente soddisfatto:  , parzialmente soddisfatto:  , e non soddisfatto: ). Di seguito, i requisiti sono raggruppati in base alla loro categoria.

15.1 Requisiti di Sicurezza

15.1.1 SR-CF-10 e SR-E-10



Per qualsiasi presentazione, il formato della Credenziale e il Trust Framework devono essere progettati in modo sicuro per determinare il Fornitore di Credenziali e verificare che la Credenziale originale sia stata emessa da questo Fornitore di Credenziali (ad esempio, utilizzando una firma crittografica).

La specifica IT-Wallet supporta sia il formato di Credenziale SD-JWT VC che mdoc-CBOR. L'autenticità e l'integrità di una Credenziale vengono verificate controllando la firma del Fornitore di Credenziali.

- Per SD-JWT, la verifica viene eseguita utilizzando l'algoritmo specificato nel parametro dell'header **alg** di SD-JWT e un riferimento verificabile alla chiave pubblica che deve essere utilizzata per la verifica della firma. Utilizzando OpenID Federation, il riferimento verificabile al materiale crittografico pubblico è l'header **kid** dell'SD-JWT, dove il materiale crittografico viene ottenuto dalla Trust Chain relativa al Credential Issuer, specificato nel claim **iss**.
- Per mdoc-CBOR, la firma del Fornitore di Credenziali è contenuta nel *Mobile Security Object* (MSO) e deve essere convalidata utilizzando la chiave pubblica del Fornitore

di Credenziali attraverso una catena di certificati attendibile contenuta nel parametro dell'header **x5chain**.

15.1.2 SR-CF-20



Per qualsiasi presentazione, il formato della Credenziale deve garantire che i dati legati alla Credenziale originale non possano essere alterati (ad esempio, utilizzando una firma crittografica).

La firma crittografica inclusa nel formato della Credenziale garantisce che qualsiasi manomissione della Credenziale comporterà una verifica fallita.

15.1.3 SR-CF-21



Per l'associazione crittografica con l'Utente, il formato di presentazione richiede al Titolare di dimostrare il possesso della chiave privata associata alla Credenziale. Questo viene tipicamente ottenuto facendo firmare al Titolare una sfida, che consiste in un valore nonce e l'identificatore univoco del Verificatore di Attestati Elettronici.

Sia SD-JWT che mdoc-CBOR supportano l'associazione crittografica con l'Utente definendo come un Titolare può presentare una Credenziale a un Verificatore di Attestati Elettronici, fornendo una prova crittografica del legittimo possesso della Credenziale.

Attualmente, per il flusso remoto, IT-Wallet supporta solo presentazioni SD-JWT. In questo scenario, il parametro KB-JWT (Key-Bound JWT) viene utilizzato per dimostrare che il Titolare possiede la chiave privata associata alla Credenziale. Il Titolare firma il KB-JWT utilizzando un **nonce** e un identificatore del Verificatore di Attestati Elettronici, utilizzando il parametro **aud**, come sfida.

15.1.4 SR-E-20



Il Trust Framework deve garantire che l'identificazione di un Fornitore di Credenziali sia unica e non ambigua. Se ci sono più istanze dello stesso Fornitore di Credenziali che utilizzano lo stesso materiale chiave, il Verificatore di Attestati Elettronici deve fidarsi di tutte le istanze allo stesso modo.

Il Trust Framework IT Wallet garantisce che ogni entità (ad esempio, un Fornitore di Credenziali) sia identificata in modo univoco attraverso chiavi crittografiche e metadati, distribuiti tramite

un attestato verificabile, come la Entity Configuration OpenID Federation verificata all'interno di una Trust Chain.

15.1.5 SR-E-30



Il modo in cui il Verificatore di Attestati Elettronici determina l'affidabilità del Fornitore di Credenziali definito nel Trust Framework deve essere protetto dall'influenza di una parte malintenzionata che può, ad esempio, introdurre entità non affidabili in una directory.

I Fornitori di Credenziali sono registrati da un Trust Anchor o dal suo Intermediario. Per verificare l'affidabilità di un Fornitore di Credenziali, un Verificatore di Attestati Elettronici deve verificare che la Trust Chain relativa al Fornitore di Credenziali sia valida e ancora attiva. Questo processo di convalida garantisce che solo entità affidabili siano autorizzate a partecipare al sistema, impedendo l'introduzione di attori non affidabili.

15.1.6 SR-E-40



Il Trust Framework deve garantire che ci sia un modo per i Verificatori di Attestati Elettronici di mantenere aggiornate le loro informazioni sui Fornitori di Credenziali affidabili e che ci sia un modo per revocare la fiducia in un Fornitore di Credenziali.

Se l'Entity Statement di un Fornitore di Credenziali viene revocato o non è disponibile, significa che il Fornitore di Credenziali non è più considerato valido all'interno della federazione. Ciò garantisce che i Verificatori di Attestati Elettronici abbiano accesso in tempo reale allo stato delle entità affidabili e possano revocare la fiducia se necessario. Tuttavia, i Verificatori di Attestati Elettronici devono controllare attivamente lo stato del Fornitore di Credenziali interrogando gli endpoint della federazione (cioè, l'endpoint fetch per ottenere la Subordinate Statement).

15.1.7 SR-I-10



Il Fornitore di Credenziali deve autenticare e identificare l'Utente correttamente secondo le aspettative del Verificatore di Attestati Elettronici (che possono essere definite in una specifica, Trust Framework o per convenzione).

Il processo di emissione utilizza flussi basati su OAuth 2.0, in particolare il flusso Authorization Code, per autenticare in modo sicuro l'Utente. Inoltre, l'autenticazione dell'Utente viene eseguita

utilizzando lo schema CieID con alto LoA, l'Autenticazione eID Substantial con Verifica MRTD oppure il PID/IT-Wallet ID.

15.1.8 SR-I-20



Il Fornitore di Credenziali deve utilizzare solo attributi corretti e aggiornati sull'Utente nella Credenziale dove sono previsti dati verificati.

Quando sono previsti dati verificati, il Fornitore di Credenziali ottiene gli attributi corretti e aggiornati dalle relative Fonti Autentiche, garantendone l'accuratezza al momento dell'emissione.

15.1.9 SR-I-30



Il Fornitore di Credenziali deve revocare una Credenziale una volta che il Fornitore di Credenziali viene a conoscenza di un potenziale abuso della Credenziale.

Il Fornitore di Credenziali è l'entità responsabile della revoca di una Credenziale. La specifica descrive diversi casi d'uso che possono attivare un processo di revoca e dettaglia il flusso di revoca in cui il Fornitore di Credenziali revoca le Credenziali su richiesta dell'Utente (attraverso l'Istanza del Wallet) dopo aver verificato il possesso delle Credenziali.

15.1.10 SR-I-40



Il Fornitore di Credenziali deve includere nella Credenziale solo dati di associazione del Titolare che sono legati all'Utente effettivo (e non, ad esempio, includere una chiave crittografica sotto il controllo di una terza parte).

Il processo di emissione lega in modo sicuro la Credenziale all'Utente come segue (vedi Fig. 12.5):

- Autorizzazione (Passi 8-10): L'Istanza del Wallet invia una richiesta di autorizzazione, e il Fornitore di Credenziali autentica l'Utente utilizzando uno schema CieID o un PID/IT-Wallet ID valido, fornendo il Token di Accesso all'Utente.
- Prova di Possesso della Chiave (Passi 12-13, 16-17): Il Wallet crea un DPoP Proof JWT, legando il Token di Accesso all'Istanza del Wallet. La stessa chiave viene poi utilizzata successivamente per richiedere la Credenziale, garantendo la continuità della proprietà.

- Emissione della Credenziale (Passi 18-21): La richiesta di Credenziale viene verificata utilizzando la prova di possesso, che è crittograficamente legata all'Utente. L'uso della stessa chiave nel DPoP garantisce che il materiale chiave sia controllato dall'Istanza del Wallet, e non da una terza parte.

15.1.11 SR-I-50



(condizionale rispetto a I-10) Il Fornitore di Credenziali deve garantire che la Credenziale sia stata memorizzata in un Wallet sicuro.

Un Wallet Attestation affidabile garantisce che l'Istanza del Wallet sia sicura e soddisfi gli standard di sicurezza richiesti prima che qualsiasi Credenziale venga emessa o memorizzata. Nei Passi 5-6 di Fig. 12.5, l'Istanza del Wallet fornisce una Wallet Attestation, che include una prova di possesso firmata con la chiave privata del Wallet. Questo attestato conferma che l'Istanza del Wallet è genuina ed è stata verificata dal Fornitore di Wallet.

Il Fornitore di Credenziali verifica questo attestato prima di consentire al Wallet di partecipare al processo di emissione, garantendo che il Wallet aderisca a specifici standard di sicurezza. Successivamente, tutte le chiavi crittografiche generate e utilizzate nel processo provengono da questa Istanza del Wallet attestata.

Nota

C'è attualmente un problema aperto su questo aspetto (https://github.com/openid/OpenID4VCI/issues/355) nella specifica OpenID4VCI.

15.1.12 SR-P-20



Il protocollo deve garantire che nessuna terza parte possa interferire con il processo di emissione in modo tale che il Fornitore di Credenziali emetta Credenziali per la terza parte all'Utente.

Questo requisito è affrontato mediante l'identificazione sicura del Fornitore di Credenziali. Il parametro ^iss^ nella risposta di autorizzazione assicura al Wallet che la risposta provenga dal Fornitore di Credenziali previsto e la verifica crittografica dei token ricevuti garantisce che siano stati emessi dal legittimo Fornitore di Credenziali. Inoltre, l'uso di PKCE evita l'iniezione del codice da un'altra sessione alla sessione dell'Utente.

15.1.13 SR-P-30



Il protocollo deve garantire che l'interazione tra il Wallet e il Verificatore di Attestati Elettronici sia protetta in modo tale che nessuna terza parte possa interferire con l'interazione modificando le informazioni trasmesse.

Il processo di presentazione avviene attraverso diversi flussi, inclusi remoto e di prossimità. Nel caso del flusso remoto, una combinazione di Request Objects firmati, utilizzo di **nonce**, validazione della Trust Chain, Wallet Attestation e associazione del Titolare garantisce che nessuna terza parte possa interferire o modificare le informazioni trasmesse tra il Wallet e il Verificatore di Attestati Elettronici. Questi meccanismi sono allineati con il Requisito di Sicurezza P-30, proteggendo l'interazione da manomissioni o attacchi di iniezione.

15.1.14 SR-P-40



Il protocollo deve garantire che l'interazione tra un attaccante e un Verificatore di Attestati Elettronici non possa essere inoltrata e completata con successo da un Utente.

Nel caso del flusso sullo stesso dispositivo, questo può essere prevenuto utilizzando e controllando correttamente il valore **nonce**, che viene creato e inviato dal Verificatore di Attestati Elettronici nella richiesta di autorizzazione. Il Verificatore di Attestati Elettronici dovrebbe mantenere una mappatura tra le sessioni dell'Utente e il **nonce** che è previsto nel flusso. Il Verificatore di Attestati Elettronici dovrebbe accettare una presentazione solo se il **nonce** nella presentazione corrisponde al **nonce** che è previsto per la sessione dell'Utente. Con questa contromisura, il Verificatore di Attestati Elettronici deve rilevare se viene inviata una presentazione che non era legata alla sessione dell'Utente o se non esiste affatto una sessione dell'Utente, prevenendo l'attacco.

Per il flusso cross-device il requisito è parzialmente soddisfatto in quanto il flusso è vulnerabile agli attacchi di Cross-Device Consent Phishing (un attaccante potrebbe iniziare il flusso di presentazione, ottenere il Request Object firmato e il codice QR, e inoltrarlo alla vittima).

Alcune misure di sicurezza sono già in atto, come l'uso di **nonce** e state. Il **nonce** garantisce la freschezza della richiesta, e lo state lega il flusso a una transazione unica, riducendo così l'opportunità per un attacco di successo.

Nota

Altre misure di sicurezza sono attualmente in fase di valutazione nell'issue numero [117](https://github.com/italia/eid-wallet-it-docs/issues/117), dove viene discussa una lista di mitigazioni da [OAuthCrossDeviceSec]. Due esempi sono:

- Codici QR a breve durata/limitati nel tempo: Ridurre la durata del codice QR (ad esempio, 2-3 minuti) è necessario per limitare la finestra temporale disponibile per gli attacchi.
- Codici QR monouso: I codici QR monouso limitano la possibilità di attacchi quando lo stesso codice QR viene inviato a più vittime.

15.1.15 SR-P-41



Il protocollo deve garantire che un attaccante non possa inoltrare con successo un'interazione tra un Wallet e un Verificatore di Attestati Elettronici a un Verificatore di Attestati Elettronici sotto il controllo dell'attaccante.

Un prerequisito per un attacco di questo tipo è che l'attaccante abbia accesso ad alcuni messaggi tra il Wallet e il Verificatore di Attestati Elettronici, ad esempio, l'attaccante potrebbe avere accesso alla presentazione contenuta nel Token VP. Dato ciò, la corretta implementazione di TLS garantisce la riservatezza, evitando la fuga della risposta. Oltre a TLS, l'implementazione esistente dei controlli **nonce** e audience nel protocollo di presentazione dovrebbe aiutare a soddisfare il requisito di sicurezza P-41. Per quanto riguarda il claim **nonce**, il Verificatore di Attestati Elettronici DEVE verificare che il valore **nonce** nel Token VP corrisponda al valore **nonce** che è creato dal Verificatore di Attestati Elettronici durante la richiesta di autorizzazione. Per quanto riguarda il valore **aud**, il Verificatore di Attestati Elettronici deve verificare che l'audience della presentazione corrisponda all'identificatore del Verificatore di Attestati Elettronici.

15.1.16 SR-P-50



Il protocollo deve garantire che terze parti non possano interferire con il processo di associazione.

Nella fase di emissione, l'associazione del Titolare avviene alla richiesta di Credenziale all'endpoint protetto della Credenziale. Ciò significa che l'attaccante deve ottenere il token di accesso prima e quindi inviare la richiesta all'endpoint della Credenziale e legare le Credenziali alle chiavi sotto il suo controllo. La specifica IT-Wallet richiede l'uso di un token di accesso

vincolato al mittente, il che significa che il token di accesso si lega al dispositivo utilizzando materiali crittografici.

La seconda superficie per l'attacco è correlata alla gestione delle chiavi. Nel caso di utilizzo di chiavi basate su software, è possibile clonare le chiavi e spostarle su un dispositivo sotto il controllo dell'attaccante, e nel caso di furto delle Credenziali, l'attaccante può facilmente creare una prova di possesso delle chiavi. IT-Wallet è meno vulnerabile a questi attacchi in quanto supporta WSCD interno locale che utilizza chiavi basate su hardware. Tuttavia, la mancanza di un profilo di certificazione che certifichi il WSCD interno locale contro attaccanti altamente capaci (la certificazione per le attuali soluzioni TEE sul mercato raggiunge al massimo AVA_VAN.3 come mostrato ad esempio in questo Rapporto di Certificazione o sito Global Platform rende il requisito solo parzialmente soddisfatto.

15.1.17 SR-V-10

(condizionale rispetto a I-50+V-20) Il Verificatore di Attestati Elettronici deve garantire che la Credenziale sia memorizzata in un Wallet sicuro.

Il Verificatore di Attestati Elettronici controlla la Wallet Attestation durante gli scambi (inviato con la risposta di autorizzazione), garantendo che soddisfi i criteri di sicurezza richiesti dal Verificatore di Attestati Elettronici e sia sotto la sola responsabilità del suo emittente, il Fornitore di Wallet affidabile.

Nota

Attualmente, non sono specificate misure di sicurezza e privacy esplicite relative a questo requisito in [OpenID4VC-SecTrust] e non è chiaramente definito cosa significhi ^memorizzato in un Wallet sicuro^. Senza questo dettaglio, questo requisito è considerato solo parzialmente soddisfatto. Infatti, la Wallet Attestation garantisce che l'Istanza del Wallet stia operando su un dispositivo sicuro e affidabile e aderisca alle rigorose politiche di sicurezza stabilite dal Fornitore di Wallet. Tuttavia, l'attestato non garantisce direttamente che ogni Credenziale all'interno del Wallet sia memorizzata in modo sicuro; verifica la sicurezza complessiva dell'ambiente del Wallet, all'interno del quale risiedono le Credenziali. Pertanto, mentre l'attestato supporta la fiducia del Verificatore di Attestati Elettronici che la Credenziale provenga da una fonte sicura, è in definitiva una garanzia ampia della sicurezza del Wallet, piuttosto che una convalida specifica della memorizzazione individuale della Credenziale.

15.1.18 SR-V-20

(condizionale rispetto a I-10) Il Verificatore di Attestati Elettronici deve garantire che la Credenziale sia stata emessa da un Fornitore di Credenziali che emette Credenziali solo a Wallet affidabili.

Controllando l'affidabilità del Fornitore di Credenziali, il Verificatore di Attestati Elettronici garantisce che la Credenziale sia stata emessa da un Fornitore di Credenziali affidabile impegnato a emettere Credenziali solo a Wallet sicuri (come per SR-I-50).

Nota

Attualmente, non sono specificate misure di sicurezza e privacy esplicite relative a questo requisito in [OpenID4VC-SecTrust], configurando questo elemento come qualcosa che richiede ulteriori sviluppi e chiarimenti.

15.1.19 SR-W-20

Il Wallet deve fornire informazioni affidabili e complete sui Fornitori di Credenziali all'Utente.

L'Istanza del Wallet scopre i Fornitori di Credenziali affidabili utilizzando risorse di terze parti affidabili, come l'API di Federazione (ad esempio, utilizzando l'Endpoint di Elenco Subordinato del Trust Anchor e dei suoi Intermediari), ispezionando i metadati del Fornitore di Credenziali e i Trust Mark per filtrare il Fornitore di PID.

Le informazioni del Fornitore di Credenziali vengono visualizzate all'Utente durante il processo di emissione e possono essere successivamente lette dall'Utente in quanto sono all'interno della Credenziale emessa. Oltre alle informazioni del Fornitore di Credenziali, il *Type Metadata* dell'Attestato Elettronico contiene anche informazioni sulla Fonte Autentica.

15.1.20 SR-W-30

Il Wallet deve fornire informazioni affidabili e complete sui Verificatori di Attestati Elettronici all'Utente.

Il Wallet convalida la Trust Chain relativa al Verificatore di Attestati Elettronici e le sue informazioni vengono visualizzate all'Utente prima della presentazione.

15.2 Requisiti di Privacy

15.2.1 PR-CF-30



Il Formato della Credenziale deve garantire che ci sia un meccanismo robusto per garantire che i dati che non devono essere rilasciati a un Verificatore di Attestati Elettronici non possano essere estratti dal Verificatore di Attestati Elettronici (Divulgazione Selettiva).

Sia SD-JWT che mdoc-CBOR forniscono la capacità di Divulgazione Selettiva, permettendo ai Titolari di rivelare solo campi specifici al Verificatore di Attestati Elettronici.

15.2.2 PR-CF-40



Il Formato della Credenziale deve supportare la protezione dalla correlazione.

Mentre la Divulgazione Selettiva è uno strumento forte per prevenire la correlazione, la completa non collegabilità non è garantita in tutti i casi. Possono sorgere problemi come la collusione del Verificatore di Attestati Elettronici o il tracciamento del Fornitore di Credenziali.

Suggerimento

L'emissione in batch, utilizzando diverse chiavi di associazione e salt per ogni Credenziale, può mitigare i rischi di non collegabilità Verificatore di Attestati Elettronici/Verificatore di Attestati Elettronici e di presentazione.

15.2.3 PR-E-60



Il Trust Framework deve garantire che il Fornitore di Credenziali non possa sapere dove l'Utente utilizza la Credenziale.

Il Verificatore di Attestati Elettronici che esegue la Trust Evaluation sul Fornitore di Credenziali di una Credenziale non deve rilasciare alcuna informazione al Fornitore di Credenziali sull'Istanza del Wallet con cui sta interagendo. Utilizzando [OID-FED] il Fornitore di Credenziali non sa chi è l'Utente che presenta la Credenziale. Inoltre, la privacy è protetta anche durante il controllo dello stato della Credenziale. Utilizzando Status List [TOKEN-STATUS-LIST], la specifica IT-Wallet garantisce che mentre il Verificatore di Attestati Elettronici controlla la validità della

Credenziale, il Fornitore di Credenziali non apprende dove o quando la Credenziale viene utilizzata.

15.2.4 PR-E-70



Il Trust Framework deve supportare la protezione dalla correlazione.

I seguenti meccanismi possono essere implementati per ridurre la correlazione:

- *Verificatore di Attestati Elettronici-Verificatore di Attestati Elettronici*: meccanismi di valutazione per garantire che un Verificatore di Attestati Elettronici richieda solo le informazioni che è autorizzato a ottenere dal Wallet. Questo approccio minimizza lo scambio di dati e aiuta a prevenire la profilazione dell'Utente attraverso potenziali collusioni tra Verificatori di Attestati Elettronici.
- *Fornitore di Credenziali-Verificatore di Attestati Elettronici*: Il Fornitore di Credenziali non richiede l'autenticazione del Verificatore di Attestati Elettronici durante la Trust Evaluation. In linea di principio, il Fornitore di Credenziali non sa quali Verificatori di Attestati Elettronici l'Utente sta accedendo e eviterà la profilazione dell'attività dell'Utente basata sull'accesso del Verificatore di Attestati Elettronici.

Suggerimento

I Trust Mark di OpenID Federation consentono la definizione di politiche personalizzate adatte alla soddisfazione di questo requisito.

15.2.5 PR-W-40



Il Wallet deve chiedere all'Utente un consenso significativo prima che una Credenziale venga utilizzata. Il Wallet deve fornire all'Utente l'opportunità di rivedere qualsiasi dato che viene rilasciato a un Verificatore di Attestati Elettronici.

Dopo aver stabilito la fiducia con il Verificatore di Attestati Elettronici, il Wallet chiede il consenso dell'Utente e fornisce all'Utente l'opportunità di rivedere e selezionare i dati da presentare al Verificatore di Attestati Elettronici.

15.2.6 PR-W-60



Il Wallet deve garantire che il Fornitore di Credenziali non possa sapere dove l'Utente utilizza la Credenziale.

Come per SR-P-80.

15.2.7 PR-W-70



Il Wallet deve garantire che il Verificatore di Attestati Elettronici non possa sapere che lo stesso Utente sta utilizzando altri Verificatori di Attestati Elettronici.

Per mitigare la collegabilità Verificatore di Attestati Elettronici/Verificatore di Attestati Elettronici per le Credenziali SD-JWT, una soluzione proposta è l'emissione in batch, che prevede l'utilizzo di diverse chiavi crittografiche utilizzate nell'associazione delle chiavi e salt per ogni Credenziale emessa. Tuttavia, l'efficacia di questi metodi non è stata ancora valutata a fondo, anche in considerazione degli impatti che questi potrebbero avere sull'esperienza dell'utente, e non è ancora disponibile per IT-Wallet.

15.3 Requisiti di Sicurezza e Privacy

15.3.1 SPR-E-50



Il Trust Framework deve garantire che i cicli di vita delle chiavi, dei certificati e delle Credenziali siano progettati in modo tale da minimizzare l'impatto di una compromissione.

Il ciclo di vita della Credenziale include un meccanismo di revoca della Credenziale basato su Status List [TOKEN-STATUS-LIST] che garantisce che le Credenziali siano correttamente revocate quando compromesse o obsolete.

La revoca di un'Entità di Federazione (cioè, Fornitore di Credenziali, Verificatore di Attestati Elettronici, Fornitore di Wallet) è invece possibile non emettendo la corrispondente Subordinate Statement su quell'Entità e impostando una breve scadenza della Trust Chain, impedendo così l'uso improprio durante la compromissione.

Suggerimento

Inoltre, [OID-FED] supporta un endpoint delle chiavi storiche per recuperare l'elenco delle chiavi scadute e revocate, con la motivazione della revoca.

15.3.2 SPR-P-10



Il protocollo deve garantire che nessuna terza parte possa leggere la Credenziale emessa dal Fornitore di Credenziali.

Per mitigare questa minaccia, la specifica IT-Wallet richiede i seguenti meccanismi di sicurezza nel processo di emissione:

- *TLS*: Utilizzato in tutte le comunicazioni tra il Wallet e il Fornitore di Credenziali, garantendo che i dati in transito siano crittografati e protetti dall'intercettazione da parte di attaccanti.
- *Wallet Attestation*: Garantisce che il Wallet operi su un dispositivo sicuro e affidabile e rispetti gli standard di sicurezza richiesti dal Fornitore di Credenziali, fornendo ulteriore garanzia che il Fornitore di Credenziali stia interagendo con un'Istanza del Wallet legittima.
- *DPoP*: Garantisce che il Titolare del token di accesso possieda la chiave privata associata ad esso, impedendo agli attaccanti di riutilizzare token intercettati.
- *Associazione del Titolare*: Lega la Credenziale a un Titolare specifico, garantendo che solo il legittimo Titolare possa utilizzare una Credenziale per autenticarsi con il Fornitore di Credenziali.
- *validazione redirect_uri*: Questa validazione garantisce che la risposta di autorizzazione sia inviata all'endpoint corretto e autorizzato, impedendo così l'intercettazione non autorizzata da parte di attori malintenzionati. Garantire l'integrità del **redirect_uri** è fondamentale per evitare qualsiasi manipolazione o reindirizzamento dell'URI.
- *PKCE*: Evita l'iniezione di un **code** di autorizzazione legittimo in un'altra sessione.

15.3.3 SPR-P-60



Il protocollo deve garantire che durante un'interazione con un Verificatore di Attestati Elettronici, un attaccante non possa leggere PII.

L'IT Wallet è conforme ai requisiti di P-60 garantendo che tutte le connessioni di rete siano protette da TLS.

Inoltre, poiché la Risposta di Autorizzazione contiene PII è importante che non venga inviata a un endpoint controllato dall'attaccante. Nella specifica IT Wallet il **response_uri**, che è l'endpoint dove il Wallet invia la Risposta di Autorizzazione, è incluso nel Request Object firmato, che viene verificato dal Wallet utilizzando la chiave pubblica del Verificatore di Attestati Elettronici e la Trust Chain. Questo garantisce che la Risposta di Autorizzazione venga inviata all'endpoint corretto. Inoltre, la Risposta di Autorizzazione è crittografata con la chiave pubblica del Verificatore di Attestati Elettronici, garantendo che solo il destinatario previsto possa decrittografare e leggere le informazioni sensibili, proteggendo ulteriormente la trasmissione.

Un altro endpoint da validare è il **redirect_uri**, che viene utilizzato per reindirizzare l'Utente al Verificatore di Attestati Elettronici dopo che la presentazione della Credenziale è completata. Nella specifica IT-Wallet, il **redirect_uri** è registrato e validato in anticipo durante l'onboarding del Verificatore di Attestati Elettronici utilizzando OpenID Federation. Durante la fase di presentazione, il Wallet deve validare questo valore verificando la fiducia con il Verificatore di Attestati Elettronici secondo la Sezione Meccanismo di Valutazione della Fiducia

Per essere sicuri che il **redirect_uri** sia ricevuto da un Wallet legittimo e non dall'attaccante, l'endpoint di risposta del Verificatore di Attestati Elettronici al ricevimento di una risposta di autorizzazione valida crea un valore crittografico fresco che è collegato alla risposta di autorizzazione e lo allega al **redirect_uri** che viene inviato al Wallet. Quando il Verificatore di Attestati Elettronici riceve il reindirizzamento, può estrarre il codice di risposta e verificare con il suo endpoint di risposta se il codice di risposta era associato a questa Risposta di Autorizzazione. (Vedi *URI di Reindirizzamento* (pagina 537)).

15.3.4 SPR-P-70



Il protocollo deve garantire che durante un'interazione con un Fornitore di Credenziali, un attaccante non possa leggere PII.

Nel processo di emissione, poiché la Credenziale all'interno di una Risposta di Credenziale contiene PII, è necessario che la Credenziale non venga inviata o intercettata da un attaccante. Per mitigare queste minacce, la specifica IT-Wallet richiede i seguenti meccanismi di sicurezza:

- *TLS*: Utilizzato in tutte le comunicazioni tra il Wallet e il Fornitore di Credenziali, garantendo che i dati in transito siano crittografati e protetti dall'intercettazione da parte di attaccanti.
- *Attestato dell'Istanza del Wallet*: Garantisce che il Wallet operi su un dispositivo sicuro e affidabile e rispetti gli standard di sicurezza richiesti dal Fornitore di Credenziali, fornendo ulteriore garanzia che il Fornitore di Credenziali stia interagendo con un'Istanza del Wallet legittima.
- *DPoP*: Garantisce che il Titolare del token di accesso possieda la chiave privata associata ad esso, impedendo agli attaccanti di riutilizzare token intercettati.
- *Associazione del Titolare*: Lega la Credenziale a un Titolare specifico, garantendo che solo il legittimo Titolare possa utilizzare una Credenziale per autenticarsi con il Fornitore di Credenziali.
- *validazione redirect_uri*: Questa validazione garantisce che la risposta di autorizzazione sia inviata all'endpoint corretto e autorizzato, impedendo così l'intercettazione non autorizzata da parte di attori malintenzionati. Garantire l'integrità del **redirect_uri** è fondamentale per evitare qualsiasi manipolazione o reindirizzamento dell'URI.
- *PKCE*: Evita l'iniezione di un **code** di autorizzazione legittimo in un'altra sessione.

Suggerimento

Un ulteriore miglioramento della sicurezza che potrebbe essere applicato per aggiungere un ulteriore livello di protezione per le informazioni sensibili dell'Utente è la crittografia delle risposte di Credenziale. Lo standard OpenID4VCI fornisce l'opzione per il Wallet di richiedere Credenziali crittografate contenenti PII includendo un oggetto **credential_response_encryption** nella sua richiesta.

Nota

Attualmente, non sono specificate misure di sicurezza e privacy esplicite relative a questo requisito in [OpenID4VC-SecTrust], rimane un lavoro in corso.

15.3.5 SPR-P-80



Il protocollo deve garantire che il Fornitore di Credenziali non possa sapere dove l'Utente utilizza la Credenziale.

Il protocollo di scambio non richiede alcuna interazione tra Verificatori di Attestati Elettronici e Fornitori di Credenziali. Inoltre, Status List che preserva la privacy, garantisce che mentre il Verificatore di Attestati Elettronici controlla la validità della Credenziale, il Fornitore di Credenziali non apprende dove o quando la Credenziale viene utilizzata.

15.3.6 SPR-W-50



Il Wallet deve garantire che le Credenziali e le chiavi private siano protette da accessi non autorizzati.

Per prevenire l'accesso non autorizzato al Wallet, viene sbloccato sul dispositivo dell'Utente inserendo un numero di identificazione personale (PIN) o utilizzando l'autenticazione biometrica, come l'impronta digitale o il riconoscimento facciale, in base alle preferenze dell'Utente e alle capacità del dispositivo. Inoltre, le chiavi crittografiche sono memorizzate in modo sicuro all'interno del WSCD, garantendo che solo l'Utente possa accedervi, impedendo così l'uso non autorizzato o la manomissione.

CAPITOLO 16

Politiche Generali di Conservazione dei Log

La conservazione dei log è un elemento chiave per garantire la sicurezza, inclusa la prevenzione delle frodi, il rilevamento degli incidenti, l'integrità del sistema e la conformità con gli obblighi legali applicabili. DEVE anche essere allineata con i requisiti definiti nella norma ISO/IEC 27001, in particolare per quanto riguarda la verificabilità, il controllo degli accessi e l'archiviazione sicura. Poiché la gestione dei log può comportare il trattamento di dati personali, costituisce anche una misura di responsabilizzazione ai sensi del GDPR, con implicazioni per la minimizzazione dei dati, la limitazione della conservazione e la limitazione delle finalità; per questi aspetti, si fa riferimento alle disposizioni pertinenti del GDPR e alle normative specifiche del settore.

Per tutto ciò che riguarda la gestione dei log, i Fornitori di Wallet, i Fornitori di Credenziali, le Fonti Autentiche e le Relying Party sono considerati Entità Organizzative.

I log relativi alle attività di scambio dati del Wallet (accessi e transazioni di emissione di Credenziali e presentazione) riguardanti l'Utente, in quanto interessato, DEVONO essere conservati per un periodo limitato per motivi di sicurezza, prevenzione delle frodi, risoluzione delle controversie e obblighi legali. I log delle Fonti Autentiche che documentano l'erogazione dei dati verso i Fornitori di Credenziali per l'emissione di Attestati Elettronici di Attributi nel Sistema IT-Wallet DEVONO essere conservati per un periodo limitato per le medesime finalità.

Le Entità Organizzative sono responsabili della conservazione dei log in base ai rispettivi ruoli. Le soluzioni relative ai Fornitori di Wallet, ai Fornitori di Credenziali, alle Fonti Autentiche e alle Relying Party DEVONO implementare la registrazione di audit per le attività degli amministratori e degli operatori di servizio con accesso ai processi di scambio dati e ai log.

A meno che specifici obblighi legali non dispongano diversamente, e con la definizione di normative specifiche di settore che definiscano motivazioni appropriate, **il periodo massimo di conservazione per i log dei dati è di 12 mesi**. I log DEVONO essere archiviati in modo sicuro per garantirne l'integrità e l'immutabilità.

16.1 Requisiti Generali di Logging ISO/IEC 27001

Le Entità Organizzative DEVONO implementare misure tecniche e organizzative appropriate in conformità con ISO/IEC 27001 per mantenere un sistema di gestione della sicurezza delle informazioni per il monitoraggio, l'audit e la risposta agli incidenti. Gli aspetti chiave della gestione dei log secondo ISO/IEC 27001 includono i seguenti requisiti:

- **A.12.4 Logging e Monitoraggio**: Le Entità Organizzative DEVONO produrre log che registrino eccezioni, guasti ed eventi di sicurezza delle informazioni. Quando necessario per motivi di sicurezza, i log DOVREBBERO acquisire le attività degli utenti con un livello di dettaglio appropriato e sufficiente per soddisfare i requisiti di sicurezza, assicurando che i dati raccolti siano pertinenti e non eccessivi. I log DOVREBBERO essere rivisti regolarmente per garantire che gli incidenti di sicurezza siano identificati e affrontati tempestivamente.
- **A.12.4.1 Registrazione degli Eventi**: Le Entità Organizzative DEVONO stabilire procedure per la registrazione degli eventi, inclusi i tipi di eventi da registrare, le informazioni da acquisire e il periodo di conservazione dei log.
- **A.12.4.2 Protezione delle Informazioni di Log**: Le Entità Organizzative DEVONO essere protette contro l'accesso non autorizzato e la manomissione. Ciò include l'implementazione di controlli di accesso, crittografia e controlli di integrità per garantire che i dati di log rimangano sicuri e affidabili.
- **A.12.4.3 Log degli Amministratori e degli Operatori**: Le attività degli amministratori di sistema e degli operatori DEVONO essere registrate e riviste regolarmente. Le Entità Organizzative DEVONO rilevare attività non autorizzate e garantire la responsabilità per le azioni intraprese sui sistemi critici.
- **A.12.4.4 Sincronizzazione degli Orologi**: Tutti i sistemi delle Entità Organizzative coinvolti nella registrazione DEVONO essere sincronizzati con una fonte temporale affidabile, garantendo timestamp accurati per correlare gli eventi tra diversi sistemi e condurre indagini forensi efficaci.
- **A.16.1.7 Gestione degli Incidenti di Sicurezza delle Informazioni**: Le Entità Organizzative DEVONO garantire che i log siano disponibili e affidabili per supportare i processi di gestione degli incidenti.

16.1.1 Politica di Conservazione dei Log del Fornitore di Wallet

Le informazioni relative alla registrazione e alla gestione delle Istanze del Wallet possono essere conservate fino a 12 mesi dopo la disattivazione/revoca del Wallet o dell'account Utente associato, a meno che obblighi legali non richiedano una conservazione più lunga.

16.1.2 Politica di Conservazione dei Log del Fornitore di Credenziali

I Fornitori di Credenziali definiscono il periodo di conservazione per le Credenziali in base alle normative specifiche del settore. In assenza di normative specifiche, il periodo di conservazione per le Credenziali NON DOVREBBE superare i 12 mesi dopo la data di scadenza, configurata al momento dell'emissione nei metadati della parte firmata dall'Emittente della Credenziale.

16.1.3 Politica di Conservazione dei Log delle Fonti Autentiche

Le Fonti Autentiche sono Entità Organizzative ai fini del presente capitolo: i log riguardano gli accessi e le transazioni su endpoint e servizi utilizzati per fornire ai Fornitori di Credenziali i dati ufficiali impiegati per la generazione degli Attestati Elettronici di Attributi nell'ambito del Sistema IT-Wallet, inclusi—ove ricorrono—gli e-service esposti tramite la Piattaforma Digitale Nazionale Dati (PDND).

Salvo obblighi di legge o normativa settoriale che dispongano diversamente, per tali log si applica il periodo massimo di conservazione di **12 mesi** stabilito nel paragrafo introduttivo. Permangono in ogni caso i vincoli del GDPR sulla minimizzazione, sulla limitazione della conservazione e sulla base giuridica del trattamento.

16.1.4 Politica di Conservazione dei Log della Relying Party

Le Relying Party POSSONO conservare i log relativi al trattamento dei dati ricevuti dal Wallet solo per la durata necessaria a fornire il servizio richiesto. Il periodo massimo di conservazione è di 24 mesi dopo la conclusione del servizio o la data di scadenza delle Credenziali presentate, a meno che obblighi legali non richiedano diversamente.

- Le Relying Party NON DOVREBBERO registrare le mappe di divulgazione della presentazione delle Credenziali, ove non necessario.

16.1.5 Funzionalità di Logging delle Istanze del Wallet

Oltre ai requisiti inclusi nel Regolamento consolidato (UE) n. 910/2014, Art 5a 4(d) e alle informazioni aggiuntive fornite nei Considerando (11) e (13), il Regolamento di esecuzione (UE) 2024/2979 della Commissione stabilisce regole dettagliate per l'applicazione del Regolamento consolidato (UE) n. 910/2014, incluso il logging nelle funzionalità principali dei Portafogli di Identità Digitale Europea.

Il Regolamento (UE) 2024/2979, Articolo 9, definisce gli obblighi di logging relativi alle Soluzioni Wallet e ai meccanismi di portabilità dei dati.

CAPITOLO 17

Termini Definiti e Riferimenti

17.1 Definizioni e Acronimi

Questa sezione mira ad uniformare la terminologia del Sistema IT-Wallet alle definizioni fornite in ARF 2.7.3 (vedere ARF Annex 1). Per ciascun termine, la definizione di IT-Wallet è stata confrontata e mappata con quella di ARF, includendo note su eventuali differenze o chiarimenti.

Le definizioni di *Utente*, *Servizio Fiduciario*, *Trust Model*, *Trusted List*, *Trust Framework*, *Attributo*, *Fornitore di Attestati Elettronici di Attributi* o *Fornitore di servizi fiduciari (TSP)*, *Person Identification Data (PID)*, *Lista di Revoca*, *Fornitore di Attestati Elettronici di Attributi qualificati* o *Fornitore di servizi fiduciari qualificati (QTSP)*, *Attestato Elettronico di Attributi (EAA)*, sono definite nel documento EIDAS-ARF.

Di seguito le descrizioni di acronimi e definizioni, correlati al presente documento utili ad approfondimenti su tematiche inerenti l' IT-Wallet e i componenti con i quali interagisce.

Access Certificate Authority
Access CA
: Autorità di Certificazione responsabile dell'emissione dei Certificati di Accesso a PID Provider, Attestation Provider e Relying Party per interazioni sicure all'interno dell'ecosistema IT-Wallet. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Asset Critici
: Asset (ad esempio, le chiavi crittografiche) la cui perdita avrebbe gravi ripercussioni sull'Istanza del Wallet. Conforme con ARF 2.7.3.

Associazione Crittografica con l'Utente

Holder Key Binding

Capacità del Titolare di dimostrare il possesso della chiave privata attestata da una terza parte di fiducia. Non presente in ARF 2.7.3.

Attestato Elettronico

Attestato

Un set firmato di Attributi in un formato specifico (ad esempio mDoc-CBOR, SD-JWT VC), può essere un PID oppure una (Q)EAA. Conforme con ARF 2.7.3.

Attestato Elettronico di Attributi

Attestato verificabile digitalmente in forma elettronica, comprovante il possesso di attributi. Conforme con ARF 2.7.3.

Attestato Elettronico di Attributi Qualificati

Attestazione verificabile digitalmente emessa da un QTSP, che comprova il possesso di attributi. Conforme con ARF 2.7.3.

Attestato Elettronico di Attributi rilasciato da o per conto di un ente pubblico

Attestato Elettronico Pubblico di Attributi

Attestato Elettronico di Attributi che contiene Attributi forniti da una Fonte Autentica pubblica. Conforme con ARF 2.7.3.

Attestato Elettronico di Interesse Pubblico

Attestato Elettronico di Attributi che contiene Attributi destinati ad attestare il rilascio, da parte dello Stato o di altre pubbliche amministrazioni, di autorizzazioni, certificazioni, abilitazioni, documenti di identità e riconoscimento, ricevute di introiti, ovvero ad assumere un valore fiduciario e di tutela della fede pubblica in seguito alla loro emissione o alle scritturazioni su di essi effettuate e, in generale, quando sono considerati carte valori ai sensi dell'articolo 2, comma 10-bis, della legge 13 luglio 1966, n. 559. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Attori Primari

Entità che realizzano le Soluzioni Tecniche per il funzionamento del Sistema IT-Wallet. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Attributi

Attributi dell'Utente

Un insieme di caratteristiche, qualità, diritti o autorizzazioni di una persona fisica o giuridica o di un oggetto o anche una sola di queste informazioni. Conforme con ARF 2.7.3.

Autenticazione

Processo elettronico che consente di confermare l'Identificazione di una persona fisica o giuridica, oppure l'origine/integrità dei dati. Conforme con ARF 2.7.3.

Autorità di Federazione

Ente di governance pubblica che emana linee guida, regole e gestisce Elenchi di Fiducia e lo stato dei partecipanti. Non presente in ARF 2.7.3.

Backend del Fornitore di Wallet

Wallet Provider Backend

Infrastruttura tecnica e componenti server gestiti da un Fornitore di Wallet. Non presente in ARF 2.7.3.

Backend della Relying Party

Infrastruttura remota composta da componenti server gestiti da un fornitore di Soluzioni di Relying Party. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Catalogo degli Attestati Elettronici

Catalogo elettronico contenente informazioni relative ai formati e agli schemi degli Attestati Elettronici, ai dati in essi contenuti e alle Fonti Autentiche. Il Catalogo contiene informazioni aggiuntive che consentono di stabilire l'autenticità e l'affidabilità delle informazioni in esso contenute. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Certificate Signing Request

Richiesta inviata a una CA contenente la chiave pubblica e le informazioni identificative utili all'emissione di un certificato digitale. Non presente in ARF 2.7.3.

Certificato di Accesso

Access Certificate

Certificato di autenticazione e convalida della Wallet Relying Party. Conforme con ARF 2.7.3.

Certificato di Registrazione

Registration Certificate

Oggetto che indica gli Attributi che la Relying Party ha registrato al fine di richiederli agli Utenti. Conforme con ARF 2.7.3.

Cryptographic Hardware Key Tag

Identificativo univoco per le Cryptographic Hardware Keys, utilizzato per accedere alla chiave privata dell'hardware. Non presente in ARF 2.7.3.

Cryptographic Hardware Keys

Coppia di chiavi generata dall'Istanza del Wallet, valida per tutta la sua durata. Non presente in ARF 2.7.3.

Dataset_id

Stringa alfanumerica definita dalla Fonte Autentica e che identifica univocamente uno specifico dataset relativo a un Attestato Elettronico di Attributi. Non presente in ARF 2. 7.3; specifico di IT-Wallet.

Dati di Identificazione Personale
Person Identification Data
PID

Insieme di dati rilasciati in conformità al diritto dell'Unione o nazionale che consentono di stabilire l'identità di una persona fisica o giuridica, o di una persona fisica che rappresenta un'altra persona fisica o giuridica. Conforme con ARF 2.7.3 / EU_2024_1183. Nelle presenti Specifiche Tecniche, i riferimenti all'emissione, alla presentazione o alla revoca di un PID denotano il PID tecnico (la struttura dati firmata che contiene i Dati di Identificazione Personale), come utilizzato nell'ARF. Il termine italiano «Attestato Elettronico di Dati di Identificazione Personale», ove impiegato come sinonimo di PID, va inteso in questo senso e **non** come Attestato Elettronico di Attributi (EAA).

Divulgazione Selettiva

Funzionalità che consente all'Utente di inviare un sottoinsieme di dati contenuti in Attestati Elettronici. Conforme con ARF 2.7.3.

eIDAS Trusted List

Archivio di informazioni sulle entità autorevoli e sul loro stato utilizzato nel quadro dei servizi fiduciari eIDAS (es. Trusted List dei PID Provider, Trusted List dei Wallet Provider, List of Trusted List che punta alle Trusted List dei QTSP e dei Fornitori di EAA degli Stati Membri, Trusted List delle CA dei Certificati di Accesso e dei Fornitori di Certificati di Registrazione). Non presente in ARF 2.7.3; specifico di IT-Wallet.

Enti Nazionali di Accreditamento

Organismi che svolgono l'attività di accreditamento su delega di uno Stato membro. Conforme con ARF 2.7.3.

Entità Organizzativa

Persona giuridica (pubblica o privata) riconosciuta per svolgere un ruolo nell'ecosistema IT-Wallet. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Federation Registry

Componente di registro dell'Infrastruttura del Registro IT-Wallet che mantiene l'elenco autorevole delle entità fidate che partecipano alla federazione, inclusi i relativi metadati di federazione, gli endpoint e le chiavi pubbliche. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Fonte Autentica

Soggetto pubblico o privato responsabile di un archivio o sistema che è considerato fonte primaria per gli Attributi o per i Dati di Identificazione Personale. Conforme con ARF 2.7.3.

Fornitore di Attestati Elettronici
Credential Issuer

Soggetto pubblico o privato che fornisce Attestati Elettronici agli Utenti (può essere un

fornitore di PID oppure un fornitore di (Q)EAA). Conforme con ARF 2.7.3; IT-Wallet aggrega sotto questo termine sia il fornitore di PID che di (Q)EAA.

Fornitore di Attestati Elettronici di Attributi
EAA Provider
Entità Organizzativa che fornisce EAAs. Conforme con ARF 2.7.3.

Fornitore di Attestati Elettronici di Attributi Qualificati
QEAA Provider
Entità Organizzativa che fornisce QEAAs. Conforme con ARF 2.7.3.

Fornitore di Attestati Elettronici di Dati di Identificazione Personale
PID Provider
Provider of Person Identification Data
Persona fisica o giuridica responsabile dell'emissione e della revoca dei Dati di Identificazione Personale e della garanzia che i Dati di Identificazione Personale di un Utente siano associati crittograficamente a una Wallet Unit. Conforme con ARF 2.7.3.

Fornitore di Certificati di Registrazione
Provider of Registration Certificates
Reg. Cert. Provider
Entità Organizzativa responsabile dell'emissione dei Certificati di Registrazione che descrivono lo stato di registrazione e i diritti di PID Provider, Attestation Provider e Relying Party. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Fornitore di Servizi
Relying Party
Wallet-Relying Party
Entità che si affida all'identificazione elettronica o al Servizio Fiduciario di un'Istanza del Wallet. Conforme con ARF 2.7.3.

Fornitore di Wallet
Entità organizzativa responsabile della gestione e della fornitura di una Soluzione Wallet. Conforme con ARF 2.7.3.

Fornitore Qualificato di Firme Elettroniche
Fornitore di Servizi Fiduciari che rilascia certificati di Firma Elettronica Qualificata. Conforme con ARF 2.7.3.

Gestore di Identità Digitale
Entità organizzativa che fornisce mezzi di identificazione elettronica preesistenti notificati utilizzati per l'autenticazione dell'Utente (ad esempio CieID / SPID), distinta dall'attestato **IT-Wallet ID** detenuto nel Wallet. Non presente in ARF 2.7.3.

Identity and Access Management
Framework per la gestione delle identità digitali e dell'accesso alle informazioni. Non presente in ARF 2.7.3.

Identity Matching

Processo con cui una Relying Party accerta che gli attributi di identificazione personale presentati in una transazione si riferiscano alla stessa persona fisica. Vedere *Identity Matching e Identity Reconciliation* (pagina 567). Non presente in ARF 2.7.3; specifico di IT-Wallet.

Identity Reconciliation

Processo con cui una Relying Party, dopo un identity matching andato a buon fine, collega quella persona fisica a una precedente sessione Utente o a un record Utente memorizzato. Vedere *Identity Matching e Identity Reconciliation* (pagina 567). Non presente in ARF 2. 7.3; specifico di IT-Wallet.

Intermediario

Intermediario di Federazione

Entità intermedia come definita nella Sezione 1.2 di OID-FED nella catena di fiducia Open ID Federation tra Trust Anchor ed entità foglia. Tale ruolo di federazione **non** coincide concettualmente con l'**Intermediario di Relying Party** previsto dal quadro europeo di identità digitale (EU_2024_1183); tuttavia, nel profilo implementativo descritto dalle presenti specifiche tecniche, un **Intermediario di Relying Party** è anche realizzato come Intermediario di Federazione (vedi **Intermediario IT-Wallet**). Conforme con ARF 2.7. 3 per le strutture di federazione.

Intermediario di Relying Party

Soggetto Aggregatore

Entità Organizzativa che agisce per conto di una o più Relying Party per fornire Soluzioni Tecniche di collegamento alle Istanze del Wallet e per l'autenticazione dell'Utente o la verifica degli Attestati Elettronici. Nel diritto dell'Unione (EU_2024_1183, articolo 5b, paragrafo 8), gli intermediari che agiscono per conto delle Relying Party sono **considerati Relying Party** ai fini della registrazione e dell'autenticazione verso i Portafogli di identità digitale europea e **non devono conservare dati sul contenuto della transazione** tra l'Utente del Portafoglio e la Relying Party intermediata. I requisiti di alto livello sulla registrazione delle Wallet Relying Party, inclusi gli intermediari, i dati minimi di registrazione, la trasparenza e i meccanismi comuni di autenticazione verso i Portafogli, sono trattati nell'Architecture and Reference Framework del Portafoglio EUDI (ARF; *Topic X – Relying Party registration*, ARF_TOPIC_X_RP). IT-Wallet disciplina l'onboarding nell'infrastruttura di fiducia nazionale, i metadati e i controlli tecnici che attuano tali obblighi insieme alle misure nazionali di esecuzione applicabili.

Intermediario IT-Wallet

Entità Organizzativa che agisce come **Intermediario di Relying Party** e che è tecnicamente realizzata come Intermediario di Federazione (OID-FED, Sezione 1.2). Pertanto, nel profilo implementativo IT-Wallet, un Intermediario di Relying Party è anche un Intermediario di Federazione: pubblica la propria Entity Configuration, emette Subordinate Statement per le Relying Party affiliate ed emette loro il Trust Mark.

Nell'infrastruttura di trust IT-Wallet, è registrato dal Trust Anchor con un `trust_mark_type` specifico, che consente all'Istanza del Wallet di identificare e mostrare all'Utente che il Relying Party richiedente opera tramite un Intermediario riconosciuto. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Istanza del Wallet
Applicazione installata sul dispositivo di un Utente, parte di un'Istanza del Wallet, che fornisce interfacce utente. Conforme con ARF 2.7.3.

Istanza di Relying Party

App di Verifica
Istanza specifica di un'applicazione oppure dispositivo in dotazione ad una Relying Party. Conforme con ARF 2.7.3.

IT-Wallet ID

Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale
Attestato Elettronico di Attributi (EAA) che contiene dati di identificazione di una persona fisica ed è rilasciato **esclusivamente per usi nazionali**. I termini **IT-Wallet ID** e **Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale** indicano il medesimo EAA nazionale. La qualificazione «di ambito nazionale» distingue questo EAA dai **Dati di Identificazione Personale (PID)** EUDI, che sono un insieme di dati ai sensi del quadro europeo di Identità Digitale e **non** costituiscono un EAA. Consente l'autenticazione e l'identificazione dell'Utente nei confronti delle Relying Party che operano nell'ambito della giurisdizione nazionale. NON DEVE essere utilizzato per interazioni cross-border e **non** costituisce un PID ai sensi di EU_2024_1183 / EU_2024/2977. NON DEVE essere confuso con il **PID** EUDI, né con un **Gestore di Identità Digitale** / schema di eID nazionale (ad esempio CieID / SPID) utilizzato solo per l'autenticazione. Il termine **EID Nazionale** NON DEVE essere usato come sinonimo di IT-Wallet ID, per evitare confusione con tali schemi. Gli identificatori tecnici (`vct` e `credential_type`) sono definiti nella sezione *Modello di Dati dell'IT-Wallet ID* (pagina 416). Non presente in ARF 2.7.3; specifico di IT-Wallet.

Key Attestation

KA
Oggetto di dati emesso da un Wallet Provider che dimostra che le chiavi utilizzate per il key binding delle credenziali risiedono in un WSCD o in un Keystore affidabile, utilizzando le API di Key Attestation (OEM). Le chiavi private vincolate a una Key Attestation DEVONO essere memorizzate nello stesso Keystore o WSCD delle chiavi degli Attestati Elettronici che la Key Attestation attesta.

Key Attestation APIs (OEM)
Meccanismo di attestazione del produttore del dispositivo che conferma se le chiavi crittografiche sono archiviate in modo sicuro in un keystore supportato dall'hardware. Esempi includono l'Android Key Attestation API per i dispositivi Android e Apple Device Check per i dispositivi iOS. Non presente in ARF 2.7.3; specifico dell'IT-Wallet.

Keystore

Keystore hardware-backed interno locale

Memorizzazione delle chiavi hardware-backed nativa del dispositivo dell'Utente, ad esempio il Trusted Execution Environment (TEE) e StrongBox su Android o il Secure Enclave su iOS. Un Keystore **non** è un WSCD nel senso certificabile High. Il profilo implementativo attuale memorizza le chiavi dell'Utente dell'Istanza del Wallet, dell'IT-Wallet ID e delle (Q)EAA in un Keystore. Non usato come termine definito in ARF 2.7.3 con questa restrizione; specifico di questo profilo implementativo.

Livello di Garanzia

Nel quadro dell'Unione per l'**identificazione elettronica**, i **livelli di garanzia** esprimono il grado di fiducia nella **correttezza dell'identificazione** delle persone fisiche o giuridiche e nella possibilità di **fare affidamento sui mezzi di identificazione elettronica**. Per gli **schemi di identificazione elettronica notificati**, EIDAS, come modificato (incluso il Quadro europeo di identità digitale codificato dal EU_2024_1183), definisce i livelli **basso (*low*)**, **sostanziale (*substantial*)** e **alto (*high*)**. Nelle presenti Specifiche Tecniche il termine **LoA** si impiega anche per i **requisiti minimi di garanzia** riguardanti l'**autenticazione dell'Utente**, il contesto dell'**Istanza del Wallet** e l'**affidabilità degli Attestati Elettronici** (inclusi i metadati di catalogo come il livello minimo di garanzia), che DEVONO restare coerenti con la normativa dell'Unione, con l'attuazione nazionale degli schemi notificati e con le discipline sui **Portafogli di Identità Digitale Europea** e sui **Person Identification Data (PID)** (ivi incluso, ove pertinente, il Regolamento di esecuzione (UE) 2024/2979 della Commissione). Non presente con questa formulazione in ARF 2.7.3; allineato al quadro normativo eIDAS / EUDI Wallet.

Member State Trusted List Provider

MS TLP

Entità Organizzativa designata da uno Stato Membro per compilare, firmare e pubblicare le Trusted List nazionali (es. Trusted List dei QTSP, Trusted List dei Fornitori di EAA) e per notificarne la posizione alla Commissione Europea. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Metadata

Artefatto digitale contenente informazioni su un'entità organizzativa (endpoint, chiavi pubbliche, ecc.). Non presente in ARF 2.7.3.

National Trust Anchor

Entità organizzativa designata a livello nazionale che agisce come radice di fiducia della federazione, operando la PKI nazionale e pubblicando le Trusted List e i metadati di federazione autorevoli per le entità subordinate (ad esempio Intermediari e Foglie). Non presente in ARF 2.7.3; specifico di IT-Wallet.

Object_id

Stringa alfanumerica definita dalla Fonte Autentica e che identifica univocamente una

specifica istanza di Attestato Elettronico di Attributi associata a un determinato Utente. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Organismo di Valutazione della Conformità
Organismo accreditato valuta e certifica le Soluzioni di Portafogli o i Fornitori di Servizi Fiduciari. Conforme con ARF 2.7.3.

Policy Language
Linguaggio formale per la definizione di policy di sicurezza, privacy e gestione dell'identità. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Processo di Accreditamento
Procedura svolta dall'Ente di Accreditamento Nazionale per accreditare gli Organismi di Valutazione della conformità (CABs), che si conclude con il rilascio di un certificato di accreditamento. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Processo di Certificazione
Procedura svolta dagli Organismi di Valutazione della conformità (CABs) per certificare le Soluzioni Wallet, che comprende anche le valutazioni tecniche periodiche. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Processo di Notifica
Procedura per l'invio delle informazioni alla Commissione Europea e l'inserimento all'interno delle Trusted List. Conforme con ARF 2.7.3.

Processo di Registrazione
Procedura per la verifica dell'idoneità e della conformità delle Entità Organizzative. Conforme con ARF 2.7.3.

Processo di Supervisione
Procedimento svolto da parte di un Organismo di Vigilanza per esaminare e garantire il corretto funzionamento del Fornitore di Wallet e di altre entità. Non presente in ARF 2. 7.3; specifico di IT-Wallet.

Pseudonimo
Identificativo alternativo anonimo che consente l'autenticazione e l'autorizzazione da parte di un entità. Conforme con ARF 2.7.3.

Registration Authority

Registrar
Soggetto responsabile della registrazione delle Entità Organizzative mediante l'emissione di Attestati di Fiducia. Conforme con ARF 2.7.3.

Registro del Sistema IT-Wallet
Registro delle entità partecipanti al Sistema IT-Wallet. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Registro delle Fonti Autentiche
Authentic Source Registry
AS Registry
: Registro contenente le Fonti Autentiche, le loro capacità dichiarate e i claim disponibili, utilizzato dai Fornitori di Attestati Elettronici per individuare e integrare fornitori di dati autorevoli. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Relazione di Fiducia
Trust Relationship
: Affidabilità tra Entità Organizzative verificata in seguito alla Trust Evaluation. Non presente in ARF 2.7.3.

Segnale (Signal Hub)
: Un segnale digitale propagato attraverso Signal Hub PDND. È utilizzato dalle Fonti Autentiche per notificare ai Fornitore di Attestati Elettronici aggiornamenti su stati e/o informazioni all'interno di un dominio gestito dalla Fonte Autentica stessa.

Self-Sovereign Identity
: Approccio che concede agli individui di avere il pieno controllo sulle informazioni relative alla propria identità digitale. Non presente in ARF 2.7.3.

Servizio di Integrità del Dispositivo
: Servizio fornito dai produttori di dispositivi per verificare l'integrità delle app e l'archiviazione sicura delle chiavi. Non presente in ARF 2.7.3.

Signal Hub
: La piattaforma PDND gestita dal Gestore PDND che abilita la raccolta e distribuzione di Segnali. Consiste di due e-Service PDND: Raccolta Segnali e Distribuzione Segnali.

Sistema IT-Wallet
: Insieme di Soluzioni Tecniche che implementano il Sistema di Wallet Digitale Italiano. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Soluzione di Relying Party
: Prodotto (software/hardware/cloud) che consente la presentazione degli Attestati Elettronici in vari contesti. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Soluzione Wallet
: Insieme di Soluzioni Tecniche al fine di garantire il corretto funzionamento delle Istanze del Wallet. Conforme con ARF 2.7.3.

Soluzioni Tecniche
: Insieme dei sistemi hardware/software e dei servizi realizzati dai Fornitori di Wallet, Fornitori di Attestati Elettronici di Dati di Identificazione Personale, ecc. Non presente in ARF 2.7.3; specifico di IT-Wallet.

Specifiche Tecniche

Specifiche che forniscono l'architettura tecnica, il quadro implementativo e i requisiti di progettazione. Conforme con ARF 2.7.3.

Titolare

Holder

Persona fisica o giuridica che riceve, gestisce e presenta Attestati Elettronici tramite l'Istanza del Wallet. Simile ad ARF 2.7.3. ARF utilizza questo termine solo nei casi di interazione tra Wallet, mentre negli altri casi utilizza il termine *Holder Wallet Unit*.

Trust

Fiducia nell'affidabilità, sicurezza e integrità delle entità e delle loro azioni. Non presente in ARF 2.7.3.

Trust Attestation

Attestazione elettronica di conformità al quadro normativo, verificabile crittograficamente. Non presente in ARF 2.7.3.

Trust Evaluation

Processo di verifica dell'affidabilità delle Entità Organizzative registrate. Non presente in ARF 2.7.3.

Trust Framework

Insieme di regole e accordi giuridicamente vincolanti per un sistema composto da più attori. Non presente in ARF 2.7.3.

Trust Layer

Componente architetturale che consente ai partecipanti di stabilire un rapporto di fiducia. Non presente in ARF 2.7.3.

Trust Model

Insieme di regole che garantiscono la legittimità dei componenti/entità nell'ecosistema IT-Wallet. Non presente in ARF 2.7.3.

Trusted List

Archivio di informazioni sugli enti autoritativi e sul loro stato. Conforme con ARF 2.7.3.

Utente

Persona fisica o giuridica che utilizza servizi fiduciari o mezzi di identificazione elettronica. Conforme con ARF 2.7.3.

Verificatore di Attestati Elettronici

Verificatore di Credenziali

Una persona o entità che utilizza un'Istanza di Relying Party. Simile ad ARF 2.7.3. ARF utilizza questo termine solo nei casi di interazione tra Wallet.

Wallet Instance Attestation
WIA
: Oggetto emesso da un Fornitore di Wallet che attesta l'integrità dell'Istanza del Wallet. Questa specifica usa **Wallet Instance Attestation (WIA)**. Il nome legacy «Wallet Attestation» NON DEVE essere usato per questo artefatto. Specifico per l'IT-Wallet.

Wallet Secure Cryptographic Application
: Applicazione che gestisce gli asset critici utilizzando le funzioni crittografiche fornite dal WSCD. Conforme con ARF 2.7.3.

Wallet Secure Cryptographic Device
: Dispositivo antimanomissione che fornisce un ambiente in cui la WSCA può proteggere gli asset critici. In questa specifica il termine **WSCD** è riservato a un dispositivo crittografico certificabile come resistente ad attaccanti con potenziale di attacco elevato (ad esempio un HSM o una smart card certificata almeno Common Criteria EAL4+ AVA_VAN.5). Allineato all'ARF; la restrizione certificabile High è specifica di questo profilo implementativo.

Wallet Unit
: Configurazione univoca di una soluzione Wallet per un singolo utente, comprensiva delle funzionalità di sicurezza. Unique configuration of a Wallet Solution for an individual User, including security features. Conforme con ARF 2.7.3.

Wallet Unit Attestation
WUA
: Attestazioni che una Wallet Unit usa durante l'emissione di un PID o di un'attestazione. Una Wallet Unit Attestation comprende due tipi: le Wallet Instance Attestation (WIA), che attestano l'integrità dell'Istanza del Wallet, e le Key Attestation (KA), che attestano la sicurezza delle chiavi crittografiche memorizzate nella Wallet Unit.

Nota

Qualora un termine non è presente nell'ARF 2.7.3, la definizione fornita in IT-Wallet è da ritenersi valida per il solo contesto italiano.

Di seguito sono riportati i principali termini e definizioni relativi agli aspetti dell'Esperienza Utente:

Authentication Button
: Pulsante che consente all'Utente di avviare il processo di Autenticazione e di utilizzare i servizi forniti dai Verificatori di Attestati Elettronici.

Brand Identity
: Insieme di elementi visivi, verbali e strategici che un servizio, un prodotto o un'entità utilizza per presentarsi all'Utente e per distinguersi dagli altri.

Brand Manual

Documento che definisce le regole e gli standard per l'utilizzo corretto e coerente degli elementi distintivi di un Brand, inclusi gli aspetti visivi, verbali e applicativi, nei diversi Touchpoint.

Call To Action

Suggerimento chiaro e diretto che incoraggia gli Utenti a intraprendere un'azione specifica. Può essere un pulsante, un link o un altro elemento che guida l'Utente verso un obiettivo specifico.

Catalogo

Funzionalità dell'Istanza del Wallet in cui viene visualizzato l'elenco di tutti gli Attestati Elettronici disponibili e ottenibili tramite l'Istanza del Wallet, e dalle quali è possibile avviare il processo di emissione.

Discovery Page

È la pagina presente nel Touchpoint della Relying Party dove l'Utente atterra per accedere alla propria area autenticata e ha lo scopo di mostrare all'Utente tutti i metodi di Autenticazione disponibili.

Engagement Button

Elemento interattivo dell'interfaccia che consente all'Utente di avviare un processo (ad esempio per autenticarsi, per richiedere il rilascio di un Attestato Elettronico, ecc.).

Esperienza Utente

L'insieme delle percezioni e delle reazioni delle persone derivanti dall'uso e/o dalle aspettative d'uso di un prodotto, sistema o servizio. In linea con la norma ISO 9241-210: 2010.

Interfaccia

L'insieme degli elementi grafici, tipografici e interattivi attraverso i quali l'Utente interagisce con il/i Touchpoint preposto/i all'erogazione di un prodotto o servizio, nel rispetto di [LG_DESIGN].

Modello di Interazione

Insieme di caratteristiche che definiscono le modalità con cui l'Utente interagisce con l'Interfaccia di uno o più Touchpoint per completare un'attività o un'operazione e conseguire un determinato scopo.

Modello di Servizio

Insieme di interazioni tra attori e Touchpoint necessari per l'erogazione e la fruizione del servizio.

Selection Page

È la pagina presente nel Touchpoint della Relying Party o delle terze parti che supportano un Credential Offer dove l'Utente viene a conoscenza di tutte le Soluzioni Wallet presenti

nel Registro del Sistema IT-Wallet e può scegliere con quale proseguire il processo di Autenticazione, presentazione o erogazione.

Touchpoint
Punto di contatto (digitale e non) tra l'Utente e il prodotto o servizio.

Trust Mark
Un elemento grafico che fornisce la prova della partecipazione degli Attori Primari al Sistema IT-Wallet e garantisce quindi il rispetto dei suoi standard.

Vista di Dettaglio
Modalità di visualizzazione estesa degli Attestati Elettronici, che mostra tutti gli Attributi contenuti.

Vista in Anteprima
Modalità di visualizzazione compatta dell'Attestato Elettronico che consente di riconoscerla e distinguerla in un elenco di Attestati Elettronici mediante la presenza di dati o elementi minimi.

Visual Identity
Insieme coerente di elementi grafici e tipografici che rappresentano visivamente un prodotto o un servizio e lo rendono distinguibile e riconoscibile.

17.1.1 Acronimi

Di seguito gli acronimi usati più di frequente nel documento:

Acronimo	Description
AAL	Authenticator Assurance Level come definito https://csrc.nist.gov/glossary/term/authenticator_assurance_level (Livello di Garanzia dell'Autenticatore)
ANPR	Anagrafe Nazionale della Popolazione Residente (Italian National Registry of the Resident Population)
API	Application Programming Interface. Insieme componenti previsti per semplificare gli scenari di integrazione di uno specifico Sistema.
ARF HLR	EUDI Wallet Architecture and Reference Framework High Level Requirements
CAB	Conformity Assessment Body (Organismo di Valutazione della Conformità)
CIE	Carta di Identità Elettronica
EAA	Electronic Attestation of Attributes (Attestato Elettronico di Attributi)
NAB	National Accreditation Body (Ente Nazionale di Accreditamento)
IAM	Identity and Access Management (Gestione dell'Identità e degli Accessi)
LoA	Level of Assurance (Livello di Garanzia)
OID4VP	OpenID for Verifiable Presentation
PDND	Piattaforma Digitale Nazionale Dati
PID	Person Identification Data (Dati di Identificazione Personale)

continues on next page

Tabella 17.1 – continua dalla pagina precedente

Acronimo	Description
PII	Personally Identifiable Information (Informazioni di Identificazione Personale)
QEAA	Qualified Electronic Attestation of Attributes (Attestato Elettronico di Attributi Qualificati)
Pub-EAA	Electronic Attestation of Attributes issued by or on behalf of a public sector body (Attestato Elettronico di Attributi rilasciato da o per conto di un ente pubblico)
SSI	Self Sovereign Identity
VC	Verifiable Credential
VP	Verifiable Presentation
WIA	Wallet Instance Attestation (Attestato dell'Istanza del Wallet)
WUA	Wallet Unit Attestation
KA	Key Attestation
WSCA	Wallet Secure Cryptographic Application (Applicazione Crittografica Sicura per il Wallet)
WSCD	Wallet Secure Cryptographic Device (Dispositivo Crittografico Sicuro per il Wallet)

17.2 Linguaggio Normativo e Convenzioni

Conformemente agli RFC 2119 e 8174 le seguenti parole chiave solamente quando appaiono con tutte le lettere in maiuscolo assumono i significati di seguito riportati:

- DEVE/DEVONO: indicano un requisito che è necessario soddisfare.
- NON DEVE/NON DEVONO: indicano un divieto assoluto.
- PUÒ/POSSONO: indicano un requisito opzionale, ovvero si può scegliere di soddisfarlo o meno senza alcun tipo di implicazione.
- DOVREBBE/DOVREBBERO/RACCOMANDATO: indicano un requisito consigliato/ raccomandato, ovvero si devono tenere in considerazione tutte le implicazioni derivanti da una eventuale scelta alternativa.
- NON DOVREBBE/NON DOVREBBERO/NON RACCOMANDATO: indicano un requisito che non è consigliato/raccomandato, ovvero si devono tenere in considerazione tutte le implicazioni derivanti dalla eventuale scelta di applicare comunque il requisito.
- OBBLIGATORIO: necessario
- OPZIONALE: facoltativo

17.3 Riferimenti Normativi

Di seguito i riferimenti normativi e i rispettivi acronimi inclusi in queste Specifiche Tecniche:

CAD	Decreto Legislativo 7 marzo 2005, n. 82, e successive modifiche, contenente il 'Codice dell'Amministrazione Digitale'.
RIF_ACCESSIBILITÀ	Linee Guida sull'Accessibilità degli strumenti informatici ai sensi dell'articolo 11 della Legge 4/2004. Direttiva (UE) 2019/882 del Parlamento europeo e del Consiglio del 17 aprile 2019 sui requisiti di accessibilità dei prodotti e dei servizi.
LG_DESIGN	Linee Guida di design per i siti internet e i servizi digitali della pubblica amministrazione, ai sensi dell'articolo 53, comma 1-ter del Decreto Legislativo 7 marzo 2005, n. 82, e successive modifiche.
EIDAS	Regolamento (UE) n. 910/2014 del Parlamento europeo e del Consiglio, del 23 luglio 2014, in materia di identificazione elettronica e servizi fiduciari per le transazioni elettroniche nel mercato interno e che abroga la direttiva 1999/93/CE (Regolamento eIDAS).

17.3.1 Risorse Ufficiali

Questa sezione contiene asset, risorse e artefatti che non possono essere distribuiti sotto forma di documentazione tecnica ma devono essere forniti nel loro formato digitale nativo. Questi includono file sorgente in formati come SVG, immagini raster, repository di codice, file di configurazione e altri asset digitali essenziali per la progettazione e l'implementazione delle Soluzioni Tecniche del Sistema IT-Wallet.

Qualsiasi risorsa tecnica aggiuntiva, repository e documentazione aggiuntiva saranno rese disponibili in questa sezione nelle versioni successive delle presenti specifiche tecniche.

Sito Ufficiale

Il sito ufficiale del Sistema IT-Wallet è disponibile all'indirizzo seguente:

- **Sito Ufficiale IT-Wallet**: https://www.wallet.gov.it

Risorse Grafiche

Di seguito sono elencate le risorse grafiche ufficiali, disponibili per il download nella cartella official_resources/ di questo repository:

- **Logo IT-Wallet**: Logo ufficiale del Sistema IT-Wallet (formato SVG) - Scarica il logo
- **Pittogramma IT-Wallet**: Pittogramma ufficiale del Sistema IT-Wallet - Scarica il pittogramma
- **IT-Wallet ID**: Elemento grafico ufficiale che assicura l'identificazione e rappresentazione univoca degli Attestati di identificazione personale (IT-Wallet ID / PID) tra tutte le Soluzioni Wallet - Scarica IT-Wallet ID
- **Trust Mark**: Elemento grafico ufficiale per l'attestazione dell'appartenenza al Sistema IT-Wallet - Scarica Trust Mark - Consulta la sezione *Trust Mark* (pagina 18);
- **Authentication Button**: Pulsante ufficiale per l'autenticazione tramite IT-Wallet - Scarica Authentication Button
- **Template di pagina**: i modelli delle pagine appartenenti al flusso di Autenticazione: Discovery page, Selection page, QR code page, Thank you page ed Error page, – Consulta Pages-templates/
- **Icone e pittogrammi**: icone e pittogrammi utili per l'implementazione del flusso di Autenticazione – Scarica OK State - Scarica KO state - Scarica order

Componenti HTML

I seguenti componenti HTML ufficiali sono disponibili nella cartella official_resources/ di questo repository:

- **Authentication Button Component**: Componente HTML/CSS ufficiale per l'autenticazione tramite IT-Wallet, basato sulla libreria Bootstrap Italia - Vedi il componente - Demo live
- **Discovery Page**: Template HTML/CSS/JS ufficiale e autoconsistente per la selezione del metodo di autenticazione (identità digitale), basato sulla libreria Bootstrap Italia - Vedi il template - Demo live

- **IT-Wallet Selection Page**: Template HTML/CSS/JS ufficiale e autoconsistente per la selezione del portafoglio digitale (wallet) con cui proseguire l'autenticazione, basato sulla libreria Bootstrap Italia - Vedi il template - Demo live
- **IT-Wallet Presentation QR Code Page**: Template HTML/CSS/JS ufficiale e autoconsistente per la pagina QR del flusso remoto di presentazione OpenID4VP (**Cross Device**), con payload dimostrativo di Authorization Request tramite custom URL scheme (`haip://`, Request Object by reference, come descritto in *Flusso Remoto* (pagina 509)). Le card della Selection Page puntano a questa demo basata sulla libreria Bootstrap Italia - Vedi il template - Demo live

Gli asset statici comuni alle tre pagine (Bootstrap Italia, CSS, JavaScript condivisi, sprite SVG, favicon e loghi) sono centralizzati in *official_resources/shared-ui/* e referenziati con percorsi relativi `../shared-ui/...`.

Brand Manual

Il Brand Manual contenente le indicazioni complete per l'utilizzo delle risorse grafiche e dell'identità visiva del Sistema IT-Wallet sarà prossimamente disponibile sul sito ufficiale.

17.4 Technical References

This section details the technical references, grouped by scope, that are part of the current implementation profile. This section includes international and national standards, draft specifications, architecture references and other relevant technical documents foundational for the interoperability and the compliance with industry standards.

17.4.1 Wallet Architecture and Reference Frameworks

EIDAS-ARF	EUDI Wallet Architecture and Reference Framework (ARF) v2.8.0.

17.4.2 EUDI Wallet Standards and Technical Specifications

EUDI-TS 3	Specification of Wallet Unit Attestations (WUA) used in issuance of PID and Attestations.
EUDI-TS 10	Data Portability and Download (Export).
EUDI-TS 12	Specification of Strong Customer Authentication (SCA) Implementation with the Wallet.

17.4.3 Infrastructure of Trust

RFC 5280	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
OID-FED-WALLET	De Marco, G., Hedberg, R., Jones, M.B., Bradley, J., Dzhuvinov, V., "Open ID Federation Wallet Architectures 1.0", October 2024, Draft 03.
OID-FED	Hedberg, R., Jones, M.B., Solberg, A.A., Bradley, J., De Marco, G., Dzhuvinov, V., "OpenID Federation 1.0", February 2026, Final.
OID-FED-SUBORDINATE-EVENTS	De Marco, G., Jones, M.B., "OpenID Federation Subordinate Events Endpoint 1.0", January 2026, Draft 00.
ETSI TS 119 412-6	ETSI TS 119 412-6 v1.1.1 - Electronic Signatures and Infrastructures (ESI); Part 6: Certificate profile requirements for PID, Wallet, EAA, QEAA, and PSBEAA providers.

17.4.4 Digital Credential Data Format

SD-JWT-VC	O. Terbu, D.Fett, B. Campbell, "SD-JWT-based Verifiable Credentials (SD-JWT VC)", October 2025, Draft 12.
ISO18013-5	ISO/IEC 18013-5 2020. Information technology — Personal identification — ISO-compliant driving license — Part 5: Mobile driving license (mDL) application.

17.4.5 Digital Credential Issuance

OpenID4VCI	Lodderstedt, T., Yasuda, K., Looker, T., Bastian, P., "OpenID for Verifiable Credential Issuance 1.0", September 2025.
OPENID4VC-HAIP	K. Yasuda, Lodderstedt, T., Bormann, C., Heenan, J., "OpenID4VC High Assurance Interoperability Profile 1.0", December 2025, Final Specification.
OAUTH-ATTESTATION-CLIENT-AUTH	Looker, T., Bastian, P., Bormann, C., "OAuth 2.0 Attestation-Based Client Authentication", September 2025, Draft 7.
OAUTH-MULT-RESP-TYPE	de Medeiros, B., Scurtescu, M., Tarjan, P., Jones, M., "OAuth 2.0 Multiple Response Type Encoding Practices", February 2014.

17.4.6 Digital Credential Presentation

OpenID4VP	Terbu, O., Lodderstedt, T., Yasuda, K., Fett, D., Heenan, J., "OpenID for Verifiable Presentations 1.0", July 2025.
OAuthCrossDeviceSec	Kasselman, P., Fett, D., Skokan, F., "Cross-Device Flows: Security Best Current Practice", July 2024, Draft 8.
OPENID4VC-HAIP	K. Yasuda, Lodderstedt, T., Bormann, C., Heenan, J., "OpenID4VC High Assurance Interoperability Profile 1.0", December 2025, Final Specification.
ISO18013-5	ISO/IEC 18013-5 2020. Information technology — Personal identification — ISO-compliant driving license — Part 5: Mobile driving license (mDL) application.
OIDC-RP-Metadata	Jones, M.B., Hedberg, R., Bradley, J., Skokan, F., "OpenID Connect Relying Party Metadata Choices 1.0", March 2026.

17.4.7 Digital Credential Revocation Check Mechanisms

TOKEN-STATUS-LIST	Looker, T., Bastian, P., Bormann, C., "Token Status List (TSL)", October 2025, Draft 13.

17.4.8 National Data Interoperability Platform Specifications

MODI	"Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni", November 2023, Version 1.2.
PDND	"Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati", June 2025, Version 2.0.
OAS3	OpenAPI 3.0 API Specification.

17.4.9 National Digital Identity Platform Specifications

SPID/CIE-OpenID-Connect-Specifications	SPID/CIE OpenID Connect.

17.4.10 Security and Protection Profiles

UNI CEI EN ISO/IEC 27001:2024	Information security, cybersecurity and privacy protection — Information security management systems — Requirement

CAPITOLO 18

Come contribuire

Il progetto IT-Wallet, incluso questo documento, segue un **processo di sviluppo aperto**. Questo approccio garantisce che il processo di sviluppo sia accessibile a tutti, invitando tutte le parti interessate a partecipare.

Di conseguenza, gli stakeholder, i membri della comunità nazionale e internazionale non sono solo incoraggiati ma anche calorosamente invitati a contribuire al perfezionamento di queste regole tecniche.

Di seguito sono riportati diversi metodi disponibili per contribuire a questo progetto:

- **GitHub issues**. Aprendo un'issue, puoi chiedere chiarimenti, proporre miglioramenti o segnalare errori editoriali. Se stai lavorando su un'issue, ti incoraggiamo ad aprire una pull request in bozza e collegarla.
- **Pull requests**. Le pull request rappresentano contributi attivi al progetto, tipicamente, ma non sempre, a seguito di discussioni basate su issue. Una volta avviata una pull request, essa facilita la discussione e la revisione delle modifiche proposte prima che vengano unite al ramo principale (*versione-corrente*).
- **Canale Slack di Developers Italia**. Slack è un'applicazione di messaggistica progettata per le aziende, che collega le persone alle informazioni di cui hanno bisogno. *Developers Italia* è una comunità aperta basata su contributi e partecipazione da parte di pubbliche amministrazioni, sviluppatori, tecnici, studenti e cittadini. *Developers Italia* ha avviato un canale Slack a cui [tutti possono unirsi gratuitamente](https://slack.developers.italia.it/), dove puoi conoscere tutte le loro attività e partecipare alle discussioni.

18.1 Ringraziamenti

Vorremmo ringraziare le seguenti persone per i loro commenti, preoccupazioni, idee, contributi, alcuni dei quali sostanziali, a questo profilo di implementazione e al set iniziale di implementazioni.

- Alen Horvat
- Amir Sharif
- Andrea Misuraca
- Andrea Moro
- Andrea Prosseda
- Chris Phillips
- Chiara Leonardi
- Elisa Nicolussi Paolaz
- Emanuele De Cupis
- Emiliano Vernini
- Francesco Grauso
- Francesco Marino
- Francesco Ventola
- Gabriella Cefalù
- Giada Sciarretta
- Gian Paolo Rossi
- Giuseppe De Marco
- Klaas Wierenga
- Kristina Yasuda
- Leif Johansson
- Lorenzo Cerini
- Lorenzo Nardi
- Manuel Rafeli
- Marco Basili
- Marco Pernpruner

- Mart Aarma
- Marta Sciunnach
- Matteo Fortini
- Michele Massetti
- Michele Silletti
- Nicola Saitto
- Niels van Dijk
- Oliver Terbu
- Paul Bastien
- Pasquale Cerqua
- Pasquale De Rose
- Peter Altmann
- Riccardo Iaconelli
- Roland Hedberg
- Rosalia Galeano
- Salvatore Laiso
- Salvatore Manfredi
- Stefania Guerra
- Stefano Alifuoco
- Tahir Ahmad
- Takahiko Kawasaki
- Thomas Chiozzi
- Torsten Lodderstedt
- Valentina Carollo
- Valerio Bozzolan
- Vladimir Duzhinov
- Zahra Ebadi Ansaroudi

Se qualcuno è stato dimenticato, vi preghiamo di accettare le nostre scuse con la richiesta di proporre la modifica di questa pagina tramite una [Pull Request](https://github.com/italia/eid-wallet-it-docs) con una breve descrizione del contributo offerto, durante quale evento o

canale, e durante quale periodo. Avremo quindi l'opportunità di scusarci nuovamente e rimediare il prima possibile, includendovi nell'elenco.

CAPITOLO 19

Rilasci Open Source

19.1 1. Principi Generali

L'ecosistema IT-Wallet è costruito su principi di apertura e trasparenza. In conformità con il **Codice dell'Amministrazione Digitale (CAD)** e con il **Regolamento europeo eIDAS 2.0**, l'adozione di framework open source, anche durante la fase di sperimentazione, è fortemente incoraggiata come fattore abilitante chiave per garantire fiducia, collaborazione, revisione tra pari e miglioramenti condivisi in tutto l'ecosistema. Questo quadro è allineato con l'**Interoperable Europe Act**, con l'obiettivo di massimizzare il valore pubblico e la sicurezza attraverso la trasparenza.

19.2 2. Requisiti Standard

Tutte le entità (Fornitori di Wallet, Fornitori di Credenziali, Relying Party) coinvolte nell'ecosistema IT-Wallet (di seguito "Proprietari di Progetto") **DOVREBBERO** aderire alle migliori pratiche del settore per garantire che il software sia utilizzabile, conforme e sicuro, seguendo almeno quelle elencate nelle sezioni seguenti.

19.2.1 2.1 Licenze

- I Proprietari di Progetto **DEVONO** utilizzare una licenza approvata dalla Open Source Initiative (OSI).
- I Proprietari di Progetto **DOVREBBERO** dare priorità alla **EUPL-1.2** (European Union Public License versione 1.2) per garantire la compatibilità legale all'interno del settore pubblico dell'UE, oppure a licenze permissive (es. Apache 2.0, MIT) per SDK e librerie al fine di massimizzarne l'adozione.

19.2.2 2.2 Gestione del Codice e Trasparenza

- **Controllo di Versione**: I Proprietari di Progetto **DEVONO** utilizzare piattaforme di hosting del codice che consentano l'accesso pubblico senza richiedere un login (es. GitHub, GitLab), garantendo una cronologia trasparente delle modifiche.
- **Documentazione**: I repository **DEVONO** includere una documentazione chiara, come:
  - `README.md`: Panoramica del progetto e istruzioni di installazione.
  - `LICENSE.md`: Termini con cui il titolare del copyright consente al destinatario di utilizzare il software (come descritto nella sezione *Licenze* sopra).
  - `CONTRIBUTING.md`: Linee guida per i contributi al codice effettuati da soggetti esterni (es. contributori della community).
- **Community**: I Proprietari di Progetto **DOVREBBERO** interagire attivamente con la community per lo sviluppo e il supporto, gestendo le issue e le richieste di contributo in modo tempestivo.

19.2.3 2.3 Sicurezza e Qualità

- **Qualità del Codice**: I Proprietari di Progetto **DOVREBBERO** implementare test automatizzati (es. test E2E e di integrazione) e pipeline di Continuous Integration. Tali pipeline **DOVREBBERO** essere pubbliche per consentirne l'ispezione.
- **Audit di Sicurezza**: I Proprietari di Progetto **DOVREBBERO** eseguire regolarmente audit del codice e analisi statiche.
- **SBOM**: I Proprietari di Progetto **DOVREBBERO** fornire una Software Bill of Materials (SBOM) per facilitare la gestione delle vulnerabilità, in linea con il **Cyber Resilience Act (CRA)**.

19.3 3. Fornitori di Wallet

19.3.1 3.1 Open Source Obbligatorio

Ai sensi del **Regolamento consolidato (UE) n. 910/2014 (eIDAS 2.0)**, Art. 5a par. 3, il codice sorgente dei componenti software applicativi delle soluzioni IT-Wallet **DEVE** essere rilasciato con licenza open-source.

Per motivi di sicurezza debitamente giustificati, il codice sorgente di componenti specifici, diversi da quelli installati sui dispositivi degli utenti, **PUÒ** rimanere chiuso, a condizione che ciò non comprometta la verificabilità complessiva della soluzione.

19.4 4. Fornitori di Credenziali e Relying Party

19.4.1 4.1 Componenti di integrazione

Per favorire l'interoperabilità dell'ecosistema e ridurre i costi di integrazione:

- I Fornitori di Credenziali e le Relying Party **DOVREBBERO** rilasciare Software Development Kit (SDK), librerie client e specifiche delle API con licenza open-source.

19.4.2 4.2 Infrastruttura di backend

I Fornitori di Credenziali e le Relying Party **DOVREBBERO** rilasciare la logica core del proprio backend con licenza open-source, a partire dalla fase di sperimentazione, per promuovere trasparenza e riuso.

19.5 5. Divulgazione Responsabile e Gestione delle Vulnerabilità

In conformità con il **Cyber Resilience Act (CRA)** e la **Direttiva NIS2 (UE) 2022/2555**:

- **Politica di Sicurezza**: Tutti i repository Open Source **DOVREBBERO** contenere un file `SECURITY.md` che descriva la procedura per la segnalazione delle vulnerabilità.
- **Segnalazione**: I Proprietari di Progetto **DOVREBBERO** istituire un canale per la divulgazione responsabile, al fine di gestire i problemi di sicurezza in modo riservato prima del rilascio pubblico.
- **Vulnerabilità sfruttate**: I Proprietari di Progetto **DEVONO** cooperare con i Computer Security Incident Response Team (CSIRT) nazionali in merito alle vulnerabilità attivamente sfruttate e seguire rigorosamente i protocolli di divulgazione coordinata per mitigare le minacce derivanti da una divulgazione irresponsabile.

CAPITOLO 20

Appendice

20.1 Istanza dell'Applicazione Mobile

Le Istanze del Wallet e delle Relying Party Mobili condividono significative somiglianze, in particolare rispetto ad alcuni aspetti relativi all'inizializzazione e alla validazione dell'integrità. Per eliminare la ridondanza, questa sezione utilizzerà il termine **Istanza dell'Applicazione Mobile** per riferirsi collettivamente a entrambe. All'interno di questo framework, il **Fornitore dell'Applicazione** assume le responsabilità del Fornitore di Wallet o del Backend della Relying Party, a seconda del contesto.

20.1.1 Inizializzazione dell'Istanza dell'Applicazione Mobile

Il flusso di Inizializzazione consente all'Istanza dell'Applicazione Mobile di registrare una coppia di chiavi a lunga durata, memorizzata in modo sicuro in un'appropriata memoria sicura all'interno del dispositivo, con il Fornitore dell'Applicazione. Questo processo avviene solo dopo che il Fornitore dell'Applicazione verifica la sicurezza e la Key Attestation emessa dal produttore del sistema operativo.

Il flusso è mostrato in *Diagramma di Sequenza dell'Inizializzazione dell'Istanza dell'Applicazione Mobile.* (pagina 706), mentre una descrizione passo-passo è fornita di seguito.

Passo 1: L'Utente avvia l'Istanza dell'Applicazione Mobile per la prima volta.

Passo 2: L'Istanza dell'Applicazione Mobile:

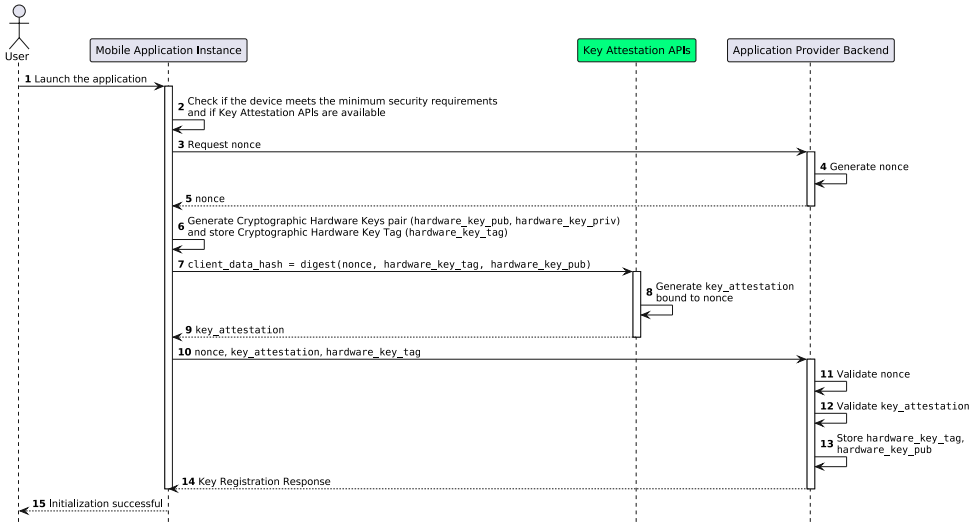


Fig. 20.1: Diagramma di Sequenza dell'Inizializzazione dell'Istanza dell'Applicazione Mobile.

- Verifica se il dispositivo soddisfa i requisiti minimi di sicurezza (*WP_021* (pagina 781)).
- Verifica se le API di Key Attestation sono disponibili (*WP_022* (pagina 781)).

Nota

Controllo della Federazione: L'Istanza dell'Applicazione Mobile deve verificare se il Fornitore dell'Applicazione fa parte della Federazione, ottenendo i suoi Metadati specifici del protocollo (*WP_023* (pagina 781)). Esempi non normativi di una risposta dall'endpoint *Endpoint di Federazione* (pagina 572) con la **Entity Configuration** e i **Metadati** del Fornitore dell'Applicazione sono presentati nelle sezioni *Entity Configuration del Fornitore di Wallet* (pagina 326) e *Entity Configuration Relying Party* (pagina 372).

Passi 3-5 (Recupero del Nonce): L'Istanza dell'Applicazione Mobile richiede un `nonce` monouso dall'**Endpoint Nonce** del Backend del Fornitore dell'Applicazione (vedi *Endpoint Nonce della Soluzione Wallet* (pagina 572) o relying-party-provider-backend-endpoint:Endpoint Nonce della Relying Party). Questo `nonce` DEVE essere imprevedibile per servire come principale difesa contro gli attacchi di replay (*WP_131* (pagina 814)).

In caso di richiesta riuscita, il Fornitore dell'Applicazione genera e restituisce il valore `nonce` all'Istanza dell'Applicazione Mobile, come parte della *Risposta di Nonce dell'Applicazione Mobile* (pagina 711). Il Fornitore dell'Applicazione DEVE garantire che sia monouso e valido solo entro un periodo di tempo specifico.

Passo 6: L'Istanza dell'Applicazione Mobile, attraverso il sistema operativo,

crea una coppia di Cryptographic Hardware Keys e memorizza il corrispondente Cryptographic Hardware Key Tag nell'archivio locale una volta soddisfatti i seguenti requisiti (*WP_132* (pagina 814)):

1. DEVE assicurarsi che le Cryptographic Hardware Keys non esistano già. Se esistono e l'Istanza dell'Applicazione è nella fase di inizializzazione, DEVONO essere eliminate.
2. DEVE generare una coppia di chiavi asimmetriche a Curva Ellittica (`hardware_key_pub`, `hardware_key_priv`) tramite un WSCD locale.
3. DOVREBBE ottenere un identificatore univoco Cryptographic Hardware Key Tag (`hardware_key_tag`) per le Cryptographic Hardware Keys generate dal sistema operativo. Se il sistema operativo consente di specificare un tag durante la creazione delle chiavi, allora DEVE essere selezionata una stringa casuale per l'`hardware_key_tag`. Questo valore casuale DEVE essere resistente alle collisioni e imprevedibile per garantire la sicurezza. Per raggiungere questo obiettivo, considerare l'utilizzo di una funzione di hash crittografico o un generatore di numeri casuali sicuro fornito dal sistema operativo o da una libreria crittografica affidabile.
4. Se i punti precedenti sono soddisfatti, DEVE memorizzare l'`hardware_key_tag` nell'archivio locale.

Nota

WSCD: L'Istanza dell'Applicazione Mobile PUÒ utilizzare un WSCD locale per operazioni crittografiche, inclusa la generazione di chiavi, l'archiviazione sicura e l'elaborazione crittografica, su dispositivi che supportano questa funzionalità. Sui dispositivi Android, Strongbox è RACCOMANDATO; Trusted Execution Environment (TEE) PUÒ essere utilizzato solo quando Strongbox non è disponibile. Per i dispositivi iOS, Secure Elements (SE) DEVONO essere utilizzati. Dato che ogni OEM offre un SDK distinto per accedere al WSCD locale, la discussione di seguito affronterà questo argomento in un contesto generale.

Se il WSCD fallisce durante una qualsiasi di queste operazioni, ad esempio a causa di limitazioni hardware, solleverà una risposta di errore all'Istanza dell'Applicazione Mobile. L'Istanza dell'Applicazione Mobile DEVE gestire questi errori di conseguenza per garantire un funzionamento sicuro. I dettagli sulla gestione degli errori sono lasciati all'implementazione dell'Istanza dell'Applicazione Mobile.

Passo 7: L'Istanza dell'Applicazione Mobile utilizza le API di Key Attestation, fornendo il `client_data_hash` per acquisire la Key Attestation (*WP_133b*

(pagina 814)).

Nota

API di Key Attestation: In questa sezione, si presume che le API di Key Attestation siano fornite dai produttori di dispositivi. Questo servizio consente la verifica di una chiave memorizzata in modo sicuro all'interno dell'hardware del dispositivo attraverso un oggetto firmato. Inoltre, offre una prova verificabile che una specifica Istanza dell'Applicazione Mobile sia autentica, inalterata e nel suo stato originale utilizzando un documento firmato specializzato creato per questo scopo.

Il servizio incorpora anche dettagli nell'oggetto firmato, come il tipo di dispositivo, il modello, la versione dell'app, la versione del sistema operativo, lo stato del bootloader e altre informazioni rilevanti per valutare se il dispositivo è stato compromesso. Inoltre, i dispositivi Android possono possedere l'*API di Key Attestation*, una funzionalità supportata da *StrongBox Keymaster* (un HSM fisico installato direttamente sulla scheda madre) o dal *TEE* (Trusted Execution Environment, un'area sicura del processore principale). *Key Attestation* mira a fornire un modo per determinare con certezza se una coppia di chiavi è supportata dall'hardware, quali sono le proprietà della chiave e quali vincoli sono applicati al suo utilizzo. Per i dispositivi Apple, l'API di Key Attestation è rappresentata da *DeviceCheck*, che fornisce un framework e un'interfaccia server per gestire i dati specifici del dispositivo in modo sicuro. *DeviceCheck* viene utilizzato in combinazione con il *Secure Enclave*, un HSM dedicato integrato nei SoC di Apple. *DeviceCheck* può essere utilizzato per attestare l'integrità del dispositivo, delle app e/o delle chiavi di crittografia generate sul dispositivo, garantendo che siano state create in un ambiente sicuro come *Secure Enclave*. Gli sviluppatori possono sfruttare la funzionalità di *DeviceCheck* utilizzando il framework stesso. Questi servizi, sviluppati specificamente dal produttore, sono integrati negli SDK Android o iOS, eliminando la necessità di un endpoint predefinito per accedervi. Inoltre, poiché sono sviluppati specificamente per l'architettura mobile, non hanno bisogno di essere registrati come Entità di Federazione attraverso i sistemi di registrazione nazionali. *Secure Enclave* è disponibile sui dispositivi Apple dall'iPhone 5s (2013). Per i dispositivi Android, l'inclusione di **Strongbox Keymaster** può variare a seconda del produttore, che decide se includerlo o meno.

Se si verificano errori nel processo delle API di Key Attestation, come la verifica dell'integrità del dispositivo, ad esempio, a causa di API di Key Attestation non disponibili, un errore interno o un nonce non valido nella richiesta di integrità, le API di Key Attestation sollevano una risposta di errore. L'Istanza dell'Applicazione Mobile DEVE elaborare questi errori di conseguenza. I dettagli sulla gestione degli

errori sono lasciati all'implementazione dell'Istanza dell'Applicazione Mobile.

Passo 8: Le API di Key Attestation eseguono le seguenti azioni:

- Creano una Key Attestation che è collegata con il `client_data_hash` fornito e la chiave pubblica dell'Hardware dell'Istanza dell'Applicazione.
- Incorporano informazioni relative alla sicurezza del dispositivo.
- Utilizzano una chiave privata OEM per firmare la Key Attestation, quindi verificabile con il relativo certificato OEM, confermando che le Cryptographic Hardware Keys sono gestite in modo sicuro dal sistema operativo.

Passo 9 (Richiesta di Inizializzazione dell'Istanza dell'Applicazione Mobile): L'Istanza dell'Applicazione Mobile invia una *Richiesta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 712) al Fornitore dell'Applicazione, per inizializzare l'Istanza dell'Applicazione Mobile, identificata dalla chiave pubblica Cryptographic Hardware. Il corpo della richiesta include i seguenti attributi: il `nonce`, la Key Attestation (`key_attestation`) e il Cryptographic Hardware Key Tag (`hardware_key_tag`) (*WP_133* (pagina 814)).

Nota

Non è necessario inviare la chiave pubblica dell'Hardware dell'Istanza dell'Applicazione perché è già inclusa nella `key_attestation`. Come visto nei passaggi precedenti, le API di Key Attestation creano una Key Attestation collegata al `client_data_hash` fornito, che è il digest del `nonce` del Fornitore dell'Applicazione, la chiave pubblica dell'Hardware dell'Istanza dell'Applicazione e il suo Hardware Key Tag (*WP_133a* (pagina 814)). Questo processo elimina la necessità di inviare direttamente la chiave pubblica dell'Hardware dell'Istanza dell'Applicazione, poiché è già inclusa nella Key Attestation.

Passi 10-12 (Risposta di Inizializzazione dell'Istanza dell'Applicazione Mobile): Il Fornitore dell'Applicazione convalida il `nonce` e la firma `key_attestation`, quindi (*WP_135–137* (pagina 814)):

1. DEVE verificare che il `nonce` sia stato generato dal Fornitore dell'Applicazione e non sia già stato utilizzato (*WP_135a* (pagina 814)).
2. DEVE convalidare la `key_attestation` come definito dalle linee guida dei produttori di dispositivi (*WP_135b* (pagina 814)). Il Fornitore dell'Applicazione DEVE anche verificare il legame tra l'`hardware_key_tag` ricevuto, l'`hardware_key_pub` e il `nonce` con il `client_data_hash` fornito nella Key Attestation (*WP_136* (pagina 814)).

3. DEVE verificare che il dispositivo in uso non abbia difetti di sicurezza e rifletta i requisiti minimi di sicurezza definiti dal Fornitore dell'Applicazione (*WP_135b* (pagina 814)).
4. Se questi controlli sono superati, DEVE registrare l'Istanza dell'Applicazione Mobile, conservando il Cryptographic Hardware Key Tag (`hardware_key_tag`), la Public Hardware Key (`hardware_key_pub`) e possibilmente altre informazioni utili relative al dispositivo (*WP_137* (pagina 814)).

In caso di inizializzazione riuscita dell'Istanza dell'Applicazione Mobile, il Fornitore dell'Applicazione risponde con una conferma di successo (*Risposta di Inizializzazione dell'Istanza dell'Applicazione Mobile* (pagina 713), *WP_135* (pagina 814)).

Nota

Il Fornitore dell'Applicazione potrebbe associare l'Istanza dell'Applicazione Mobile (attraverso l'identificatore `hardware_key_tag`) a un Utente o Dispositivo specifico (*WP_138* (pagina 814)). Questo identifica in modo univoco l'Utente/Dispositivo all'interno dei sistemi del Fornitore dell'Applicazione e può essere utilizzato per future revoche nel ciclo di vita dell'Istanza dell'Applicazione Mobile.

Passi 13-14: L'Istanza dell'Applicazione Mobile è stata inizializzata.

Nota

Modello di Minaccia: mentre l'endpoint di inizializzazione non necessita di autenticare il client, è protetto attraverso l'uso di *key_attestation*. La corretta validazione di questa attestazione permette l'inizializzazione di istanze di app autentiche e inalterate. Qualsiasi altro attributo inviato non sarà sottoposto a validazione, portando l'endpoint a rispondere con un errore. Inoltre, l'inclusione di un nonce aiuta a prevenire gli attacchi di replay. L'autenticità sia del nonce che dell'`hardware_key_tag` è garantita dalla firma trovata all'interno della `key_attestation`.

Richiesta di Nonce dell'Applicazione Mobile

La Richiesta di Nonce utilizza il metodo HTTP GET.

Di seguito è riportato un esempio non normativo di una Richiesta di Nonce.

```
GET /nonce HTTP/1.1
Host: application-provider.example.com
```

Risposta di Nonce dell'Applicazione Mobile

In caso di richiesta riuscita, il Fornitore dell'Applicazione restituisce una Risposta HTTP con un codice di stato `200 OK`, con `Content-Type` impostato su `application/json`.

Il corpo della Risposta di Nonce contiene il valore `nonce`.

Di seguito è riportato un esempio non normativo di una Risposta di Nonce.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "nonce": "d2JhY2NhbG91cmVqdWFuZGFt"
}
```

Risposta di Errore di Nonce dell'Applicazione Mobile

Se si verificano errori, il Fornitore dell'Applicazione restituisce una risposta di errore. La risposta utilizza `application/json` come `Content-Type` e include i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato.

Di seguito è riportato un esempio non normativo di una Risposta di Errore di Nonce.

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "error": "server_error",
    "error_description": "The server encountered an unexpected error."
}
```

La seguente tabella elenca i Codici di Stato HTTP e i relativi codici di errore supportati per la risposta di errore:

Codice di Stato HTTP	Codice di Errore	Descrizione
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint Nonce ha riscontrato un problema interno.

continues on next page

Tabella 20.1 – continua dalla pagina precedente

Codice di Stato HTTP	Codice di Errore	Descrizione
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint Nonce è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

Richiesta di Inizializzazione dell'Istanza dell'Applicazione Mobile

La Richiesta di Inizializzazione dell'Istanza utilizza il metodo HTTP POST con `Content-Type` impostato su `application/json`.

Il corpo della Richiesta di Inizializzazione dell'Istanza contiene i seguenti attributi:

Attributo	Descrizione	Riferimento
nonce	DEVE essere impostato sul valore ottenuto dal Fornitore dell'Applicazione attraverso l'Endpoint Nonce.	Questa specifica.
hardware_key_tag	L'identificatore univoco delle **Cryptographic Hardware Keys** e codificato in `base64url`.	Questa specifica.
key_attestation	Un'attestazione che garantisce la generazione, l'archiviazione e l'utilizzo sicuri della coppia di chiavi generata dall'Istanza dell'Applicazione Mobile. Questo può essere un array contenente una catena di certificati il cui certificato foglia è la Key Attestation ottenuta dalle **API di Key Attestation** del dispositivo, firmata con la chiave hardware del dispositivo.	Questa specifica.

Di seguito è riportato un esempio non normativo di una Richiesta di Inizializzazione dell'Istanza.

```
POST /instance-initialization HTTP/1.1
Host: application-provider.example.com
Content-Type: application/json

{
  "nonce": "d2JhY2NhbG91cmVqdWFuZGFt",
  "key_attestation": "o2NmbXRvYXBwbGUtYXBw... redacted",
  "hardware_key_tag": "WQhyDymFKsP95iFqpzdEDWW4l7aVna2Fn4JCeWHYtbU="
}
```

Risposta di Inizializzazione dell'Istanza dell'Applicazione Mobile

Se una Richiesta di Inizializzazione dell'Istanza viene convalidata con successo, il Fornitore dell'Applicazione fornisce una Risposta HTTP con codice di stato `204 No Content`.

Di seguito è riportato un esempio non normativo di una Risposta di Inizializzazione dell'Istanza.

```
HTTP/1.1 204 No content
```

Risposta di Errore di Inizializzazione dell'Istanza dell'Applicazione Mobile

Se si verificano errori, il Fornitore dell'Applicazione restituisce una risposta di errore. La risposta utilizza `application/json` come `Content-Type` e include i seguenti parametri:

- *error*. Il codice di errore.
- *error_description*. Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore riscontrato (*WP_035* (pagina 781)).

Di seguito è riportato un esempio non normativo di una Risposta di Errore di Inizializzazione dell'Istanza.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store

{
    "error": "forbidden",
    "error_description": "The provided nonce is invalid, expired, or already used. "
}
```

La seguente tabella elenca i Codici di Stato HTTP e i relativi codici di errore supportati per la risposta di errore (*WP_036–040* (pagina 781)):

Codice di Stato HTTP	Codice di Errore	Descrizione
`400 Bad Request`	`bad_request`	La richiesta è malformata, mancano parametri richiesti o include parametri non validi e sconosciuti.
`403 Forbidden`	`integrity_check_error`	Il dispositivo non soddisfa i requisiti minimi di sicurezza del Fornitore dell'Applicazione.
`403 Forbidden`	`invalid_request`	Il nonce fornito non è valido, è scaduto o è già stato utilizzato.

continues on next page

Tabella 20.3 – continua dalla pagina precedente

Codice di Stato HTTP	Codice di Errore	Descrizione
`403 Forbidden`	`invalid_request`	La firma della Key Attestation non è valida.
`422 Unprocessable Content` [OPZIONALE]	`validation_error`	La richiesta non aderisce al formato richiesto.
`500 Internal Server Error`	`server_error`	Si è verificato un errore interno durante l'elaborazione della richiesta.
`503 Service Unavailable`	`temporarily_unavailable`	Il servizio non è disponibile. Si prega di riprovare più tardi.

20.2 e-Service PDND

Il framework EIDAS-ARF consente agli Stati Membri di stabilire le interfacce, i termini e le condizioni che regolano la comunicazione tra i Fornitori di Credenziali e le Fonti Autentiche. Nel contesto italiano, l'interoperabilità è stabilita sfruttando le seguenti linee guida:

- "Linee Guida sull'interoperabilità tecnica delle Pubbliche Amministrazioni" (MODI);
- "Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati" (PDND).

Per utilizzare la PDND, le entità DEVONO aderire formalmente, diventando **Aderenti**. All'interno dell'infrastruttura PDND, gli Aderenti DEVONO assumere almeno uno dei seguenti ruoli:

- **Erogatori**: espongono e-Service ad altri Aderenti.
- **Fruitori**: utilizzano e-Service offerti dagli Erogatori all'interno dell'Infrastruttura PDND.

L'accesso ad un e-Service richiede ai Fruitori di ottenere uno specifico Access Token, noto all'interno dell'infrastruttura PDND come Voucher.

20.2.1 Requisiti e Pattern di Sicurezza

Questa specifica si basa sul seguente insieme di requisiti:

ID	Descrizione	Tipo
R1	Il Fruitore e l'Erogatore hanno entrambi aderito all'infrastruttura PDND.	Architetturale
R2	La comunicazione tra il Fruitore e l'Erogatore DEVE garantire l'integrità dei dati, l'autenticità, la non ripudiabilità e la protezione contro attacchi di replay.	Sicurezza
R3	L'Erogatore PUÒ richiedere al Fruitore di includere dati tracciati nella richiesta. In tal caso, DEVE esserci una correlazione tra i dati tracciati e il Voucher.	Sicurezza
R4	Gli e-Service DEVONO essere implementati in REST, quindi il protocollo SOAP NON DEVE essere utilizzato.	Tecnico
R5	L'Erogatore DEVE garantire, con un alto grado di certezza, la prova di possesso del Voucher da parte del Fruitore.	Sicurezza

PDND e MODI definiscono diversi pattern di sicurezza progettati per migliorare specifiche proprietà di sicurezza nelle interazioni tra gli Aderenti. Questa specifica adotta i seguenti pattern di sicurezza:

Pattern di Sicurezza	Conforme a
[REST_JWS_2021_POP] Profilo di emissione dei Voucher JWS POP (*Allegato 3 - Standard e dettagli tecnici utilizzati per la fruizione dei Voucher di autorizzazione* [PDND]): OBBLIGATORIO. Aggiunge una prova di possesso sul Voucher. Il Fruitore che utilizza il Voucher per accedere a un e-Service DEVE dimostrare la prova di possesso della chiave privata la cui chiave pubblica è attestata nel Voucher.	R2, R4, R5
[ID_AUTH_CHANNEL_01] Direct Trust Transport-Level Security (*Allegato 2 - Pattern di sicurezza* [MODI]): OBBLIGATORIO. Protegge la comunicazione tra il Fruitore e l'Erogatore garantendo riservatezza, integrità, identificazione dell'Erogatore e mitigazione contro attacchi di replay e spoofing.	R1, R2
[INTEGRITY_REST_02] Integrità del payload delle richieste REST in PDND (*Allegato 2 - Pattern di sicurezza* [MODI]): CONDIZIONALE. Garantisce l'integrità del payload della richiesta REST del Fruitore e della risposta REST dell'Erogatore, all'interno dell'Infrastruttura PDND. È OBBLIGATORIO ogni volta che la richiesta/risposta contiene un payload.	R2, R4

continues on next page

Tabella 20.5 – continua dalla pagina precedente

Pattern di Sicurezza	Conforme a
[AUDIT_REST_02] Inoltro dati tracciati nel dominio del Fruitore REST con correlazione (*Allegato 2 - Pattern di sicurezza* [MODI]): OPZIONALE. L'Erogatore PUÒ richiedere dati aggiuntivi tracciati nel dominio del Fruitore, con una correlazione tra tali dati e il metodo di autenticazione. In tal caso, questo pattern DEVE essere utilizzato.	R3, R4
[ID_AUTH_REST_01] Direct Trust su PDND tramite token JWT emessi dalla piattaforma (*Allegato 2 - Pattern di sicurezza* [MODI]).	R1, R2, R4, R5

Nota

In queste specifiche, il pattern di sicurezza `REST_JWS_2021_POP` è implementato in conformità a **RFC 9449**.

In alternativa, la prova di possesso PUÒ essere attestata dal JWT `Tracking Evidence` (come dettagliato di seguito). Tuttavia, mentre il `TrackingEvidence` è definito in `AUDIT_REST_02` per fornire dati tracciati aggiuntivi, in questo contesto funge da prova di possesso del Voucher. Tali scelte di implementazione saranno indicate rispettivamente come `POP_DPoP` e `POP_TPoP`.

I seguenti pattern di sicurezza definiti in PDND e MODI NON DEVONO essere utilizzati in quanto non conformi ai requisiti definiti in precedenza:

- I seguenti pattern possono essere utilizzati solo quando il Fruitore non può iscriversi all'Infrastruttura PDND (ossia quando la trust tra gli Aderenti deve essere stabilita in forma diretta), pertanto non risultano conformi a **R1**:
  - **[ID_AUTH_CHANNEL_02]** Direct Trust mutual Transport-Level Security (*Allegato 2 - Pattern di Sicurezza* [MODI])
  - **[ID_AUTH_REST_02]** Direct Trust su PDND tramite token JWT emessi dalla piattaforma con caratteristiche di unicità del token/messaggio (*Allegato 2 - Pattern di sicurezza* [MODI]).
  - **[INTEGRITY_REST_01]** Integrità del payload messaggio REST (*Allegato 2 - Pattern di sicurezza* [MODI]).
- Il seguente pattern non fornisce correlazione tra i dati tracciati e il Voucher, pertanto non risulta conforme a **R3**:
  - **[AUDIT_REST_01]** Inoltro dati tracciati nel dominio del Fruitore REST (*Allegato 2 - Pattern di sicurezza* [MODI]).

- I seguenti pattern sono basati su un'architettura SOAP, pertanto non risultano conformi a **R4**:
  - **[ID_AUTH_SOAP_01]** Direct Trust con certificato X.509 su SOAP (*Allegato 2 - Pattern di sicurezza* [MODI]).
  - **[ID_AUTH_SOAP_02]** Direct Trust con certificato X.509 su SOAP con unicità del token/messaggio (*Allegato 2 - Pattern di sicurezza* [MODI]).
  - **[INTEGRITY_SOAP_01]** Integrità del payload del messaggio SOAP (*Allegato 2 - Pattern di sicurezza* [MODI]).
- Il seguente pattern non garantisce la prova di possesso del Voucher, pertanto non risulta conforme a **R5**:
  - **[REST_JWS_2021_Bearer]** Profilo di emissione dei Voucher JWS Bearer (*Allegato 3 - Standard e dettagli tecnici utilizzati per la fruizione dei Voucher di autorizzazione* [PDND]).

Nota

Nel caso di implementazione `POP_TPoP`, il Voucher viene emesso come token Bearer. Tuttavia, poiché è accompagnato da una prova di possesso, è comunque conforme al pattern di sicurezza `REST_JWS_2021_POP` invece che `REST_JWS_2021_Bearer`.

20.2.2 Emissione del Voucher PDND

L'Infrastruttura PDND definisce due diversi tipi di Voucher:

- **Per e-Service**: consentono ai Fruitori di richiedere dati da un e-Service.
- **Per API di Interoperabilità**: consentono agli Aderenti di richiedere dati dall'API di Interoperabilità, esposta dall'Infrastruttura PDND.

I due flussi sono descritti di seguito.

Voucher PDND per e-Service

Prerequisiti per il Voucher PDND per e-Service

Il **Fruitore** DEVE rispettare i seguenti prerequisiti:

- Ha completato con successo l'adesione all'Infrastruttura PDND (come da R1).

- Ha creato un nuovo *Client e-service* per interagire con l'e-Service desiderato. Al momento della creazione, gli è stato assegnato un `client_id` dalla Piattaforma PDND.
- Ha registrato una coppia di chiavi associata al *Client e-service*.
- Ha richiesto di iscriversi all'e-Service desiderato.
- Ha definito un nuovo scopo per l'e-Service. Al momento della definizione, gli è stato assegnato un `purposeId` dalla Piattaforma PDND.
- Ha associato il *Client e-service* allo scopo definito.

L'**Erogatore** DEVE rispettare i seguenti prerequisiti:

- Ha completato con successo l'adesione all'Infrastruttura PDND (come da R1).
- Ha creato un nuovo e-Service e lo ha pubblicato all'interno del Catalogo API PDND.
- Ha approvato la richiesta del Fruitore di iscriversi all'e-Service.

Flusso del Voucher PDND per e-Service

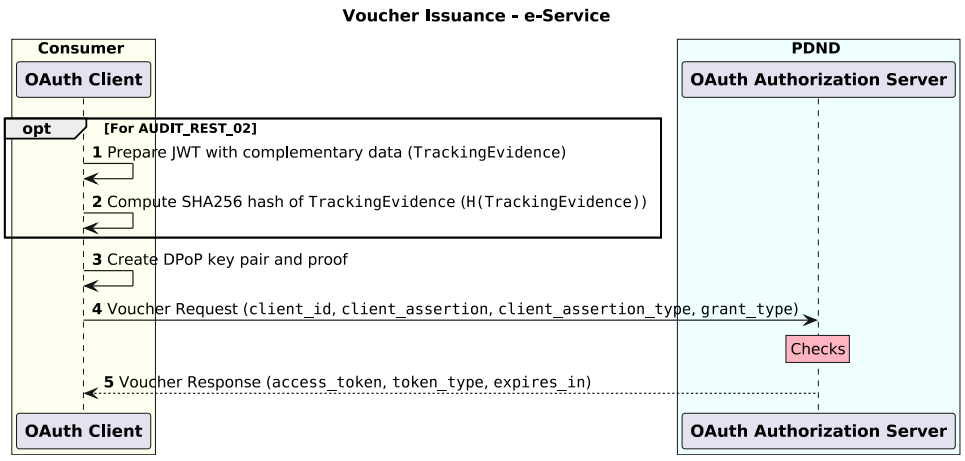


Fig. 20.2: Emissione del Voucher per e-Service - Flusso dettagliato.

Passi 1-2 (Preparazione dei dati tracciati): Il Fruitore prepara un JWT (`TrackingEvidence`) contenente i dati tracciati che devono essere inviati all'Erogatore. Infine, calcola l'hash SHA-256 di `TrackingEvidence`.

```
{
  "alg": "ES256",
  "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
  "typ": "JWT"
}
```

```
{
  "iss": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
  "aud": "https://erogatore.example/ente-example/v1",
  "exp": 1733052600,
  "nbf": 1733036450,
  "iat": 1733036400,
  "jti": "a4b5c6d7-e8f9-abcd-ef12-345678901234",
  "dnonce": 6528424213685,
  "purposeId": "b2c3d4e5-f6g7-h8i9-j0k1-lmno12345678",
  "userID": "a8b7c6d5-e4f3-g2h1-i9j0-klmnopqrstuv",
  "loa": "substantial"
}
```


Nota

I passi 1-2 sono richiesti solo quando si rispetta il pattern di sicurezza `AUDIT_REST_02` o l'implementazione `POP_TPoP`.

Passo 3 (Generazione coppia di chiavi e 'DPoP proof'): Il Fruitore DEVE creare una nuova coppia di chiavi e un nuovo JWT di *DPoP proof*, seguendo le istruzioni fornite nella Sezione 4 di **RFC 9449** per la richiesta di token all'Authorization Server PDND.

Nota

Il passo 3 è richiesto solo quando si segue l'implementazione `POP_DPoP`.

Passo 4 (Richiesta di Voucher): Il Fruitore crea una *Voucher Request* e la invia all'Authorization Server PDND.

```
POST /authorization-server/token HTTP/1.1
Host: interop.pagopa.it
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwia2V5X29wcyI6WyJz
aWduIl0sImtpZCI6IjM5ZmE5NjBiLTc3M2YtNDllZi04YTBlLWU3NzNlOWI5N2FlOCIsImNydiI6IlAtMjU2IiwieC
I6Imh1eVhJUU52OTAyb0xzcFg0X3pvbkM5NEc2eUVsbjZsc2RtLTF3TTczMm8iLCJ5IjoiSTlQREVhd1dIcWFGREd4
MVprTmstMlBWNldkcGNhSDNBZk9iQlNMaWhndyJ9fQ.eyJqdGkiOiI1NTBlODQwMC1lMjliLTQxZDQtYTcxNi00NDY
2NTU0NDAwMDAiLCJodG0iOiJQT1NUIiwiaHR1IjoiaHR0cHM6Ly9pbnRlcm9wLnBhZ29wYS5pdC9hdXRob3JpemF0a
W9uLXNlcnZlciIsImlhdCI6MTc2MjI2MDYxNn0.D0ZUDkfGHx_rQBdYi_3VSXkdbJM-7FSWN88LWICQImMWtIWd2mb
xGb7v8udfM_c4_ase8x7I3I1JZm01Us3QEA
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_id=82914b3f-60b2-4529-b4d6-3d4e67f0a933&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2N
TQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdW
IiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJpbnRlcm9wLnBhZ29wYS5pdC9h
dXRob3JpemF0aW9uLXNlcnZlciIsImV4cCI6MTczMzA0MTQ0MCwiaWF0IjoxNzMzMDM3ODQwLCJqdGkiOiI3ZTlmM2
E0ZC1jOWIyLTQyZjYtYTZkNC0zOGUxMmZiNmI4YWIiLCJwdXJwb3NlSWQiOiJkMmI5YTY1My1jNDk3LTQ1YzYtYjhm
MS01YmRmMTI0YzlkM2EiLCJkaWdlc3QiOnsiYWxnIjoiU0hBMjU2IiwidmFsdWUiOiI5Yzc4OTRhMGE1YTkxMDU4MG
I5NjdmMzg0Y2RmYmExN2IxYWI2Zjg2NjcwZTViMGRmMThhMGM0NTNiNWViMjE1In19.cl-wvwJF3WLgywoq9qULVKC
ajleqz0jpD82QTZZAxHSjoGeA7l7n0LNC8eDfIM4F-rzMU5qfC9eW6UDxMwJrdg

{
    "alg": "ES256",
    "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
    "typ": "JWT"
}
```

```
{
    "iss": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "sub": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "aud": "https://interop.pagopa.it/authorization-server",
    "exp": 1733041440,
    "iat": 1733037840,
    "jti": "7e9f3a4d-c9b2-42f6-a6d4-38e12fb6b8ab",
    "purposeId": "d2b9a653-c497-45c6-b8f1-5bdf124c9d3a",
    "digest": {
        "alg": "SHA256",
        "value": "9c7894a0a5a910580b967f384cdfba17b1ab6f86670e5b0df18a0c453b5eb215"
    }
}
```


Nota

Il claim `purposeId` nel payload di `client_assertion` è richiesto solo quando si richiede un Voucher per e-Service.

Nota

Il claim `digest` nel payload di `client_assertion` è richiesto solo quando si rispetta il pattern di sicurezza `AUDIT_REST_02` o l'implementazione `POP_TPoP`.

Alla ricezione della *Voucher Request*, l'Authorization Server PDND DEVE eseguire i seguenti controlli sui parametri del relativo body:

- Il claim `client_assertion_type` è impostato su `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- Il claim `grant_type` è impostato su `client_credentials`.

L'Authorization Server PDND DEVE anche validare il JWT `client_assertion` come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `JWT`.

Firma:

- Ottenere la chiave pubblica del Fruitore corrispondente al parametro header `kid`, interagendo con l'API di Interoperabilità PDND.
- Validare la firma del JWT utilizzando la chiave pubblica del Fruitore recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- I claim `iss` e `sub` DEVONO identificare un Client registrato nell'Infrastruttura PDND.
- Il claim `aud` DEVE rappresentare l'Authorization Server PDND.
- Il claim `exp` DEVE rappresentare un istante temporale successivo all'ora corrente.
- Se il claim `nbf` è presente, DEVE rappresentare un istante temporale precedente all'ora corrente.
- Il claim `iat` DEVE rappresentare un istante temporale precedente all'ora corrente.
- Il claim `jti` NON DEVE essere stato utilizzato in precedenza.
- Il claim `purposeId` DEVE identificare uno scopo registrato nell'Infrastruttura PDND e associato al Client.

Nota

L'Authorization Server PDND non deve eseguire alcun controllo sul claim `digest`.

Nota

La verifica dei claim `exp`, `nbf`, `iat` e `jti`, come dettagliato sopra, DEVE essere eseguita per tutti i JWT descritti in questa sezione. Questi controlli non saranno esplicitamente menzionati nei riferimenti successivi.

Passo 6 (Emissione del Voucher): Qualora i controlli abbiano successo, l'Authorization Server PDND emette un Voucher, che è incluso nella *Voucher Response* inviata al Fruitore.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "access_token": "eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN
2YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjgyOTE0YjNmLT
YwYjItNDUyOS1iNGQ2LTNkNGU2N2YwYTkzMyIsImF1ZCI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1l
eGFtcGxlL3YxIiwiZXhwIjoxNzMzMDQyMTUwLCJuYmYiOjE3MzMwNDE5NDUsImlhdCI6MTczMzA0MTkyMCwianRpIj
oiYzRmNWQ3ZTItYjdjOC00MGY2LTliNmEtZGM5YTRmNWFlYjU3IiwiY2xpZW50X2lkIjoiODI5MTRiM2YtNjBiMi00
NTI5LWI0ZDYtM2Q0ZTY3ZjBhOTMzIiwicHVycG9zZUlkIjoiZDJiOWE2NTMtYzQ5Ny00NWM2LWI4ZjEtNWJkZjEyNG
M5ZDNhIiwiZGlnZXN0Ijp7ImFsZyI6IlNIQTI1NiIsInZhbHVlIjoiOTkwOGQ5NGI4ZmViMjY4YzAzNzEwNmQ3Yzg5
ZTcwNjBjMmNjMWY2YjJiNGViY2I4MDViZmVlNTNhNTM5MzA3YiJ9LCJjbmYiOnsiamt0IjoiMFpjT0NPUlpOWXktRF
dwcXEzMGpaeUpHSFROMGQySGdsQlYzdWlndUE0SSJ9fQ.sGhaHEOfMTB7r4_8ZILM_a9eTBGawWn3kL-dxYoZggFIz
yrXDOZcQWt0zr00lMk2iYAMWxS32e4cUedmAsBXGw",
    "token_type": "DPoP",
    "expires_in": 3600
}

{
    "alg": "ES256",
    "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
    "typ": "at+jwt"
}

{
    "iss": "https://interop.pagopa.it",
    "sub": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "aud": "https://erogatore.example/ente-example/v1",
    "exp": 1733042150,
    "nbf": 1733041945,
    "iat": 1733041920,
    "jti": "c4f5d7e2-b7c8-40f6-9b6a-dc9a4f5aeb57",
    "client_id": "82914b3f-60b2-4529-b4d6-3d4e67f0a933",
    "purposeId": "d2b9a653-c497-45c6-b8f1-5bdf124c9d3a",
    "digest": {
        "alg": "SHA256",
        "value": "9c7894a0a5a910580b967f384cdfba17b1ab6f86670e5b0df18a0c453b5eb215"
    },
    "cnf": {
        "jkt": "0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
    }
}
```

Nota

Il claim `digest` nel payload di `access_token` è richiesto solo quando si rispetta

il pattern di sicurezza `AUDIT_REST_02` o l'implementazione `POP_TPoP`. Se presente, corrisponde al valore del claim `digest` contenuto in `client_assertion`.

Voucher PDND per API di Interoperabilità

Prerequisiti per il Voucher PDND per API di Interoperabilità

L'**Aderente** DEVE rispettare i seguenti prerequisiti:

- Ha completato con successo l'adesione all'Infrastruttura PDND.
- Ha creato un nuovo *Client interop api* per interagire con l'API di Interoperabilità. Al momento della creazione, gli è stato assegnato un `client_id` dalla Piattaforma PDND.
- Ha registrato una coppia di chiavi associata al *Client interop api*.

Flusso del Voucher per API di Interoperabilità

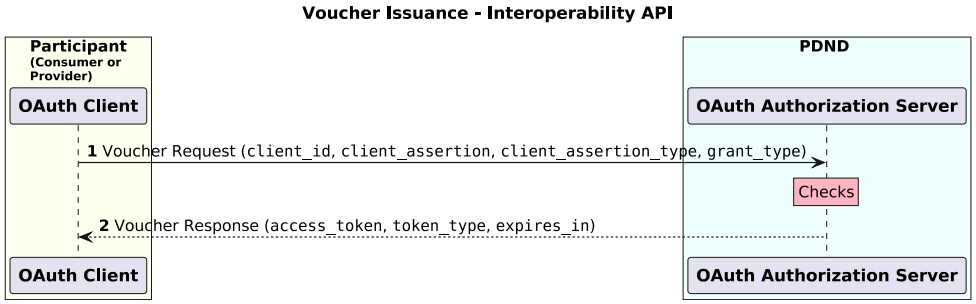


Fig. 20.3: Emissione del Voucher per API di Interoperabilità - Flusso dettagliato.

Passo 1 (Richiesta di Voucher): L'Aderente crea una *Voucher Request* e la invia all'Authorization Server PDND.

```
POST /authorization-server/token HTTP/1.1
Host: interop.pagopa.it
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_id=5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjlhNGQ4ZTNmLThiN2QtNGM5OC05MjZmLTI3NDVjNmIxZ
jgzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI1YTNjN2YyOC05MWI5LTRjNGUtODlhOS02ZTJmODVkOTI2MmIiLCJzdW
IiOiI1YTNjN2YyOC05MWI5LTRjNGUtODlhOS02ZTJmODVkOTI2MmIiLCJhdWQiOiJpbnRlcm9wLnBhZ29wYS5pdC9h
dXRob3JpemF0aW9uLXNlcnZlciIsImV4cCI6MTczMzIzMzUwMCwiaWF0IjoxNzMzMjMyMzAwLCJqdGkiOiJkMmM5YT
diNC0zZTgxLTRkMjctYjZmNy01MWE4YzlmMGEzYzYifQ.YDX7ekvvY3gPHTfZeqa3IcurU7kNBZPy3OHAdljdXSFLo
C5cVVyIzl43aMbwLouI43ylxWktaf0-pXabmye1qA

{
    "alg": "ES256",
    "kid": "9a4d8e3f-8b7d-4c98-926f-2745c6b1f832",
     "typ": "JWT"
}

{
    "iss": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
    "sub": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
    "aud": "https://interop.pagopa.it/authorization-server",
    "exp": 1733233500,
    "iat": 1733232300,
    "jti": "d2c9a7b4-3e81-4d27-b6f7-51a8c9f0a3c6"
}
```

Alla ricezione della *Voucher Request*, l'Authorization Server PDND DEVE eseguire i seguenti controlli sui parametri del relativo body:

- Il claim `client_assertion_type` è impostato su `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
- Il claim `grant_type` è impostato su `client_credentials`.

L'Authorization Server PDND DEVE anche validare il JWT `client_assertion` come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `JWT`.

Firma:

- Ottenere la chiave pubblica dell'Aderente corrispondente al parametro header `kid`, interagendo con l'API di Interoperabilità PDND.

- Validare la firma del JWT utilizzando la chiave pubblica dell'Aderente recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- I claim `iss` e `sub` DEVONO identificare un Client registrato nell'Infrastruttura PDND.
- Il claim `aud` DEVE rappresentare l'Authorization Server PDND.

Passo 2 (Emissione del Voucher): Qualora i controlli abbiano successo, l'Authorization Server PDND emette un Voucher, che è incluso nella *Voucher Response* inviata all'Aderente.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN2Y
wOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjVhM2M3ZjI4LTkx
YjktNGM0ZS04OWE5LTZlMmY4NWQ5MjYyYiIsImF1ZCI6Imh0dHBzOi8vaW50ZXJvcC5wYWdvcGEuaXQvYXBpL3YxIi
wiZXhwIjoxNzMzMjM2NjgwLCJuYmYiOjE3MzMyMzMxNTgsImlhdCI6MTczMzIzMzA4MCwianRpIjoiZjg3ZTJkNWIt
OWY2NS00ZjBmLThhZDQtOTJlNThlNmIxM2M3IiwiY2xpZW50X2lkIjoiNWEzYzdmMjgtOTFiOS00YzRlLTg5YTktNm
UyZjg1ZDkyNjJiIn0.SKDDap16Ubi6gYwpKVdBcuhmhF_XnGiHeoxkF8F4IAualYORu_TxnDZqeP_RCcBAxSRkJTFb
MihPCLA7DoRQOw",
  "token_type": "Bearer",
  "expires_in": 3600
}

{
  "alg": "ES256",
  "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
  "typ": "at+jwt"
}
```

```
{
  "iss": "https://interop.pagopa.it",
  "sub": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b",
  "aud": "https://interop.pagopa.it/api/v1",
  "exp": 1733236680,
  "nbf": 1733233158,
  "iat": 1733233080,
  "jti": "f87e2d5b-9f65-4f0f-8ad4-92e58e6b13c7",
  "client_id": "5a3c7f28-91b9-4c4e-89a9-6e2f85d9262b"
}
```


Endpoint Authorization Server PDND

L'Endpoint Authorization Server PDND emette Voucher agli Aderenti. Questi Voucher consentono ai Fruitori di accedere alle risorse degli e-Service e agli Erogatori di interagire con l'API di Interoperabilità.

Richiesta (Voucher PDND)

La richiesta all'Endpoint Authorization Server PDND aderisce al flusso Client Credentials Grant specificato in **RFC 6749**. Il client si autentica presentando un'asserzione client basata su JWT come definito in **RFC 7521** e **RFC 7523**.

Seguendo le suddette specifiche, la richiesta DEVE essere una richiesta HTTP POST con un corpo codificato in formato `application/x-www-form-urlencoded`.

La *Voucher Request* DEVE includere i seguenti parametri nell'header HTTP (se non diversamente specificato):

Parametro	Descrizione	Riferimento
DPoP	JWT di *DPoP proof*, per rispettare il pattern di sicurezza `REST_JWS_2021_POP`. È obbligatorio solo se il Voucher richiesto è per e-Service (quindi non per API di Interoperabilità) e segue l'implementazione `POP_DPoP`.	[**RFC 9449**], [PDND]

La *Voucher Request* DEVE includere i seguenti parametri nel body:

Parametro	Descrizione	Riferimento
client_id	Identificativo univoco del Client dell'Aderente, assegnato dalla PDND.	[**RFC 6749**], [**RFC 7521**], [**RFC 7523**], [PDND]
client_assertion	Un JWT che rappresenta l'asserzione del client.	[**RFC 7521**], [**RFC 7523**], [PDND]
client_assertion_type	DEVE essere impostato su `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.	[**RFC 7521**], [**RFC 7523**]
grant_type	DEVE essere impostato su `client_credentials`.	[**RFC 6749**], [**RFC 7523**]

Il JWT `client_assertion` DEVE includere i seguenti parametri nel JOSE header:

Parametro	Descrizione	Riferimento
alg	Identificativo di un algoritmo di firma digitale.	[**RFC 7515**]
kid	Identificativo univoco del JWK utilizzato dall'Aderente per firmare `client_assertion`.	[**RFC 7515**]
typ	DEVE essere impostato su `JWT`.	[**RFC 7515**], [**RFC 7519**]

Il JWT `client_assertion` DEVE includere i seguenti claim nel payload (se non diversamente specificato):

Claim	Descrizione	Riferimento
iss	DEVE essere impostato sullo stesso valore di `client_id`.	[**RFC 7523**]
sub	DEVE essere impostato sullo stesso valore di `client_id`.	[**RFC 7523**]
aud	Identificativo dell'Endpoint Authorization Server PDND.	[**RFC 7523**]
exp	Timestamp UNIX che rappresenta l'istante di scadenza del JWT.	[**RFC 7523**]
nbf	Timestamp UNIX che rappresenta il primo istante di validità del JWT (opzionale).	[**RFC 7519**]
iat	Timestamp UNIX che rappresenta l'istante di emissione del JWT.	[**RFC 7523**]
jti	Identificativo univoco del JWT per prevenire attacchi di replay.	[**RFC 7523**]

continues on next page

Tabella 20.8 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
purposeId	Identificativo dello scopo registrato nella Piattaforma PDND, associato all'e-Service previsto. È obbligatorio solo se il Voucher richiesto è per e-Service (quindi non per API di Interoperabilità).	[MODI], [PDND]
digest	Oggetto JSON contenente il digest del JWT `Tracking Evidence`. È obbligatorio solo se il Voucher richiesto è per e-Service (quindi non per API di Interoperabilità), e quando si rispetta `AUDIT_REST_02` o l'implementazione `POP_TPoP`. Se presente, DEVE contenere i seguenti claim: • **alg**: stringa JSON che rappresenta l'algoritmo di hashing; • **value**: stringa JSON che rappresenta il valore del digest.	[MODI]

Risposta (Voucher PDND)

La *Voucher Response* DEVE includere i seguenti parametri nel body:

Parametro	Descrizione	Riferimento
access_token	JWT che rappresenta l'Access Token emesso dall'Endpoint Authorization Server PDND.	[**RFC 6749**], [**RFC 9449**], [PDND]
token_type	DEVE essere impostato su: • `DPoP` in caso di Voucher per e-Service seguendo l'implementazione `POP_DPoP`; • `Bearer` in caso di Voucher per API di Interoperabilità, o Voucher per e-Service seguendo l'implementazione `POP_TPoP`.	[**RFC 6749**], [**RFC 9449**]
expires_in	Numero che rappresenta la durata dell'Access Token in secondi come intero positivo.	[**RFC 6749**], [**RFC 9449**]

Il JWT `access_token` DEVE includere i seguenti parametri nel JOSE header:

Parametro	Descrizione	Riferimento
alg	Identificativo di un algoritmo di firma digitale.	[**RFC 7515**]

continues on next page

Tabella 20.10 – continua dalla pagina precedente

Parametro	Descrizione	Riferimento
kid	Identificativo univoco del JWK utilizzato dall'Endpoint Authorization Server PDND per firmare `access_token`.	[**RFC 7515**]
typ	DEVE essere impostato su `at+jwt`.	[**RFC 9068**]

Il JWT `access_token` DEVE includere i seguenti claim nel payload (se non diversamente specificato):

Claim	Descrizione	Riferimento
iss	Identificativo dell'Authorization Server PDND.	[**RFC 7519**], [**RFC 9068**]
sub	Identificativo dell'Aderente, corrispondente al parametro `client_id` nel body della *Voucher Request*.	[**RFC 7519**], [**RFC 9068**]
aud	Identificativo dell'e-Service.	[**RFC 7519**], [**RFC 9068**]
exp	Timestamp UNIX che rappresenta l'istante di scadenza del JWT.	[**RFC 7519**], [**RFC 9068**]
nbf	Timestamp UNIX che rappresenta il primo istante di validità del JWT (opzionale).	[**RFC 7519**]
iat	Timestamp UNIX che rappresenta l'istante di emissione del JWT.	[**RFC 7519**], [**RFC 9068**]
jti	Identificativo univoco del JWT per prevenire attacchi di replay.	[**RFC 7519**], [**RFC 9068**]
client_id	DEVE corrispondere al `client_id` contenuto nella *Voucher Request*.	[**RFC 7519**], [**RFC 8963**], [**RFC 9068**], [PDND]
purposeId	DEVE corrispondere al valore del claim `purposeId` contenuto nella *Voucher Request*. È obbligatorio solo se il Voucher richiesto è per e-Service (cioè, non per API di Interoperabilità).	[MODI], [PDND]
digest	DEVE corrispondere al valore dell'oggetto `digest` contenuto nella *Voucher Request*. È obbligatorio solo quando si rispetta `AUDIT_REST_02` o l'implementazione `POP_TPoP`.	[MODI]

continues on next page

Tabella 20.11 – continua dalla pagina precedente

Claim	Descrizione	Riferimento
cnf	DEVE contenere un claim **jkt** che è il JWK SHA-256 Thumbprint Confirmation Method. Il valore del claim *jkt* DEVE essere la codifica base64url (come definito in [RFC 7515]) del JWK SHA-256 Thumbprint della chiave pubblica DPoP (in formato JWK) a cui l'Access Token è associato. È obbligatorio solo quando si rispetta l'implementazione `POP_DPoP`.	[RFC 9449. Sezione 6.1] e [RFC 7638].

Se si verificano errori durante la validazione della *Voucher Request*, l'Endpoint Authorization Server PDND DEVE restituire una risposta di errore come definito in RFC 6749#section-5.2. La risposta DEVE utilizzare `application/json` come `Content-Type` e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "invalid_request",
  "error_description": "The client_assertion_type parameter is missing."
}
```

La seguente tabella elenca gli HTTP Status Code e i relativi codici di errore che DEVONO essere supportati per la risposta di errore:

HTTP Status Code	Codice di Errore	Descrizione
`400 Bad Request`	`invalid_request`	La richiesta non può essere soddisfatta perché mancano parametri richiesti, contiene parametri non validi o è in qualche modo malformata [RFC 6749#section-5.2].
`400 Bad Request`	`invalid_grant`	La richiesta non può essere soddisfatta perché il grant fornito (cioè, `client_assertion`) è scaduto, revocato, già utilizzato o in qualche modo malformato [RFC 6749#section-5.2].
`400 Bad Request`	`unsupported_grant_type`	La richiesta non può essere soddisfatta perché il tipo di grant fornito non è supportato dall'Authorization Server PDND [RFC 6749#section-5.2].

continues on next page

Tabella 20.12 – continua dalla pagina precedente

HTTP Status Code	Codice di Errore	Descrizione
`400 Bad Request`	`invalid_scope`	La richiesta non può essere soddisfatta perché il `purposeId` fornito non è valido, è sconosciuto, malformato o non associato al Client [**RFC 6749#section-5.2**].
`400 Bad Request`	`invalid_dpop_proof`	La richiesta non può essere soddisfatta perché contiene una *DPoP proof* non valida [**RFC 9449#section-5**].
`401 Unauthorized`	`invalid_client`	La richiesta non può essere soddisfatta perché l'autenticazione del Client è fallita (quindi `client_assertion` è malformato, firmato in modo errato, mancante o non verificabile) [**RFC 6749#section-5.2**].
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Authorization Server PDND ha riscontrato un problema interno.
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Authorization Server PDND è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

20.2.3 Reperimento delle Chiavi

Chiavi dell'Authorization Server PDND

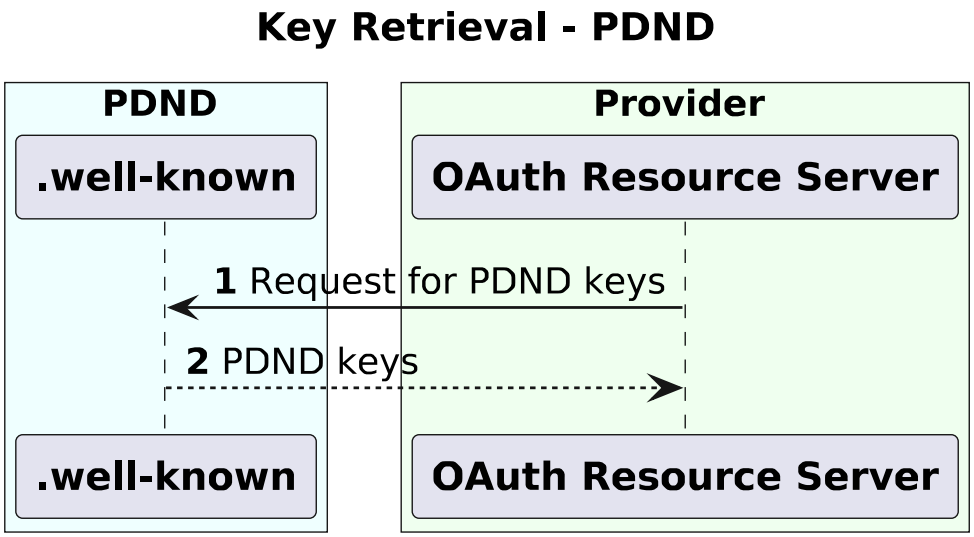


Fig. 20.4: Reperimento delle chiavi dell'Authorization Server PDND - Flusso dettagliato.

Passo 1 (Richiesta delle chiavi): L'Erogatore richiede le chiavi utilizzate dalla PDND per firmare i Voucher.

```
GET /.well-known/jwks.json HTTP/1.1
Host: interop.pagopa.it
```

Passo 2 (Risposta): L'Endpoint .well-known restituisce l'elenco delle chiavi utilizzate dalla PDND per firmare i Voucher, come un `JWK Set` [**RFC 7517**].

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "keys": [
    {
      "kty": "RSA",
      "n": "qU2Bp7xgkXBQI2w2PZ5LZGo34TIjoir-ul0x4jZ_d9hN6q...",
      "e": "AQAB",
      "alg": "ES256",
      "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec"
    },
    {
      "kty": "RSA",
      "n": "05VukHBwiE1W_kgUS0zkOyHCrRivgw5cfSTmcvD_phieEY...",
      "e": "AQAB",
      "alg": "ES256",
      "kid": "9432c16b-7aae-49df-b9c4-ea61b556652b"
    }
  ]
}
```

Endpoint .well-known dell'Authorization Server PDND

L'Endpoint .well-known fa parte dell'Infrastruttura PDND ed è utilizzato per reperire le chiavi pubbliche utilizzate dall'Authorization Server PDND per firmare i Voucher.

Richiesta (Chiavi dell'Authorization Server PDND)

La *Keys Request* è una richiesta HTTP `GET` inviata all'Endpoint .well-known. Questo endpoint consente agli Aderenti di reperire le chiavi pubbliche necessarie per verificare le firme digitali sui Voucher emessi dall'Authorization Server PDND.

Risposta (Chiavi dell'Authorization Server PDND)

L'Endpoint .well-known risponde con un HTTP Status Code `200 OK` e un `JWK Set` [**RFC 7517**] contenente le chiavi pubbliche impiegate dall'Authorization Server PDND per firmare i Voucher.

Se si verificano errori durante il reperimento delle chiavi, l'Endpoint .well-known DEVE restituire una risposta di errore. La risposta DEVE utilizzare `application/json` come `Content-Type` e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
  "error": "server_error",
  "error_description": "The server encountered an unexpected error."
}
```

La seguente tabella elenca gli HTTP Status Code e i relativi codici di errore che DEVONO essere supportati per la risposta di errore:

HTTP Status Code	Codice di Errore	Descrizione
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint .well-known ha riscontrato un problema interno.
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint .well-known è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

Chiavi degli Aderenti

Prerequisiti per il Reperimento delle Chiavi degli Aderenti

L'**Aderente** che richiede la chiave DEVE rispettare i seguenti prerequisiti:

- Ha completato con successo l'adesione all'Infrastruttura PDND (come da R1).

- Ha creato un nuovo *Client api interop* per interagire con l'API di Interoperabilità PDND. Al momento della creazione, gli è stato assegnato un `client_id` dalla Piattaforma PDND.
- Ha registrato una coppia di chiavi associata al *Client api interop*.
- Ha ottenuto un Voucher valido per interrogare l'API di Interoperabilità PDND, relativo allo specifico *Client api interop*.

Flusso di Reperimento delle Chiavi degli Aderenti

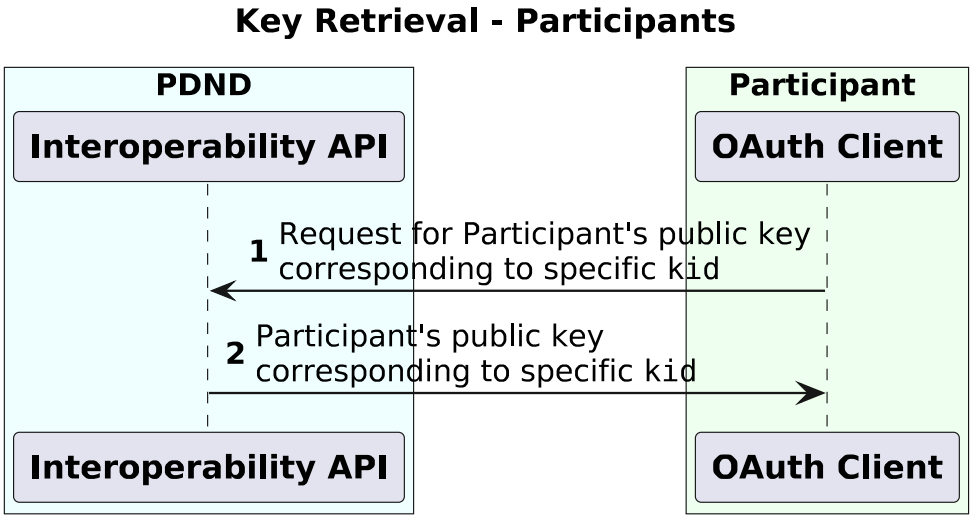


Fig. 20.5: Reperimento delle chiavi degli Aderenti - Flusso dettagliato.

Passo 1 (Richiesta della chiave): L'Aderente richiede la chiave utilizzata da un altro Aderente, corrispondente a uno specifico `kid`, alle API di Interoperabilità PDND.

```
GET /keys/c7e3d6a4-5b99-4298-9b84-d8f3a61279f1 HTTP/1.1
Host: interop.pagopa.it
Authorization: Bearer eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNi
N2YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjVhM2M3ZjI4L
TkxYjktNGM0ZS04OWE5LTZlMmY4NWQ5MjYyYiIsImF1ZCI6Imh0dHBzOi8vaW50ZXJvcC5wYWdvcGEuaXQvYXBpL3Y
xIiwiZXhwIjoxNzMzMjM2NjgwLCJuYmYiOjE3MzMyMzMxNTgsImlhdCI6MTczMzIzMzA4MCwianRpIjoiZjg3ZTJkN
WItOWY2NS00ZjBmLThhZDQtOTJlNThlNmIxM2M3IiwiY2xpZW50X2lkIjoiNWEzYzdmMjgtOTFiOS00YzRlLTg5YTk
tNmUyZjg1ZDkyNjJiIn0.SKDDap16Ubi6gYwpKVdBcuhmhF_XnGiHeoxkF8F4IAualYORu_TxnDZqeP_RCcBAxSRkJ
TFbMihPCLA7DoRQOw
```

Passo 2 (Risposta): L'Endpoint API di Interoperabilità restituisce la chiave richiesta, come un `JWK` [**RFC 7517**].

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "kty": "EC",
  "key_ops": [
    "sign"
  ],
  "kid": "b839f4c7-1e5d-4a8a-9fc6-72d3b7f091ec",
  "crv": "P-256",
  "x": "huyXIQNv902oLspX4_zonC94G6yEln6lsdm-1wM732o",
  "y": "I9PDEawWHqaFDGx1ZkNk-2PV6WdpcaH3AfObBSLihgw"
}
```

Nota

L'API di Interoperabilità include un endpoint di notifica degli eventi che avvisa gli Aderenti iscritti sui cambiamenti all'interno dell'Infrastruttura PDND. Tra queste notifiche, l'endpoint `/events/keys` fornisce aggiornamenti sulle modifiche al materiale crittografico, come aggiunte o eliminazioni di chiavi. Sfruttando questo meccanismo, gli Aderenti possono implementare una strategia di polling periodico per recuperare tutte le chiavi modificate e aggiornare la loro cache locale. Ciò elimina la necessità di richiedere ogni chiave individualmente durante il flusso.

Endpoint API di Interoperabilità PDND

L'Endpoint API di Interoperabilità fa parte dell'Infrastruttura PDND ed è utilizzato per recuperare le chiavi pubbliche di altri Aderenti all'Infrastruttura PDND.

Richiesta (Chiave da API di Interoperabilità PDND)

La *Key Request* è una richiesta HTTP `GET` inviata all'API `/keys/<kid>`. Questa richiesta viene utilizzata per recuperare una chiave specifica identificata dal suo `kid` univoco.

La *Key Request* DEVE includere i seguenti parametri di header HTTP:

Parametro	Descrizione	Riferimento
Authorization	Voucher rilasciato dall'Authorization Server PDND.	[RFC 9449]

Risposta (Chiave da API di Interoperabilità PDND)

Nel caso in cui esista una chiave pubblica con il `kid` fornito, l'Endpoint dell'API di Interoperabilità risponde con un codice di stato `200 OK` e un `JWK` [RFC 7517] che rappresenta quella chiave.

Se si verificano errori durante il reperimento della chiave, l'Endpoint API di Interoperabilità DEVE restituire un errore, la cui struttura dipende dalla natura dell'errore.

In caso di problemi di autenticazione (cioè, Voucher non valido o scaduto), la risposta DEVE aderire al formato di errore definito in RFC 6750#section-3, con specifico riferimento all'uso del parametro di header `WWW-Authenticate`.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token", error_description="The access token expire
d"
```

Per tutti gli altri errori, la risposta DEVE aderire al formato di errore definito in RFC 6749#section-5.2. La risposta DEVE utilizzare `application/json` come tipo di contenuto e DEVE includere i seguenti parametri:

- `error`: Il codice di errore.
- `error_description`: Testo in forma leggibile dall'uomo che fornisce ulteriori dettagli per chiarire la natura dell'errore incontrato.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "invalid_request",
    "error_description": "The kid parameter is missing."
}
```

La seguente tabella elenca gli HTTP Status Code e i relativi codici di errore che DEVONO essere supportati per la risposta di errore:

HTTP Status Code	Codice di Errore	Descrizione
`400 Bad Request`	`invalid_request`	La richiesta non può essere soddisfatta perché mancano parametri richiesti, contiene parametri non validi o è in qualche modo malformata [RFC 6750#section-3.1].
`401 Unauthorized`	`invalid_token`	La richiesta non può essere soddisfatta perché il Voucher è scaduto, revocato, malformato o in qualche modo non valido [RFC 6750#section-3.1].

continues on next page

Tabella 20.14 – continua dalla pagina precedente

HTTP Status Code	Codice di Errore	Descrizione
`404 Not Found`	`not_found`	La richiesta non può essere soddisfatta perché non è stata trovata alcuna chiave pubblica corrispondente al `kid` fornito.
`500 Internal Server Error`	`server_error`	La richiesta non può essere soddisfatta perché l'Endpoint API di Interoperabilità ha riscontrato un problema interno.
`503 Service Unavailable`	`temporarily_unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint API di Interoperabilità è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

20.2.4 Utilizzo dell'e-Service

Prerequisiti per l'Utilizzo dell'e-Service

Il **Fruitore** DEVE rispettare i seguenti prerequisiti:

- Ha ottenuto un Voucher valido per interagire con l'e-Service desiderato, relativo a uno specifico *Client e-service*.

L'**Erogatore** DEVE rispettare i seguenti prerequisiti:

- Ha creato un nuovo portachiavi associato allo specifico e-Service.
- Ha registrato una coppia di chiavi associata al portachiavi.

Nota

Il portachiavi dell'Erogatore è la controparte del Client relativo al Fruitore. Memorizza materiale crittografico, consentendo ai Fruitori di verificare l'integrità delle risposte dagli Erogatori.

Flusso di Utilizzo dell'e-Service

Passo 1 (Preparazione della firma): Il Fruitore prepara un JWT (`Signature`) contenente le intestazioni firmate del messaggio, per garantire l'integrità.

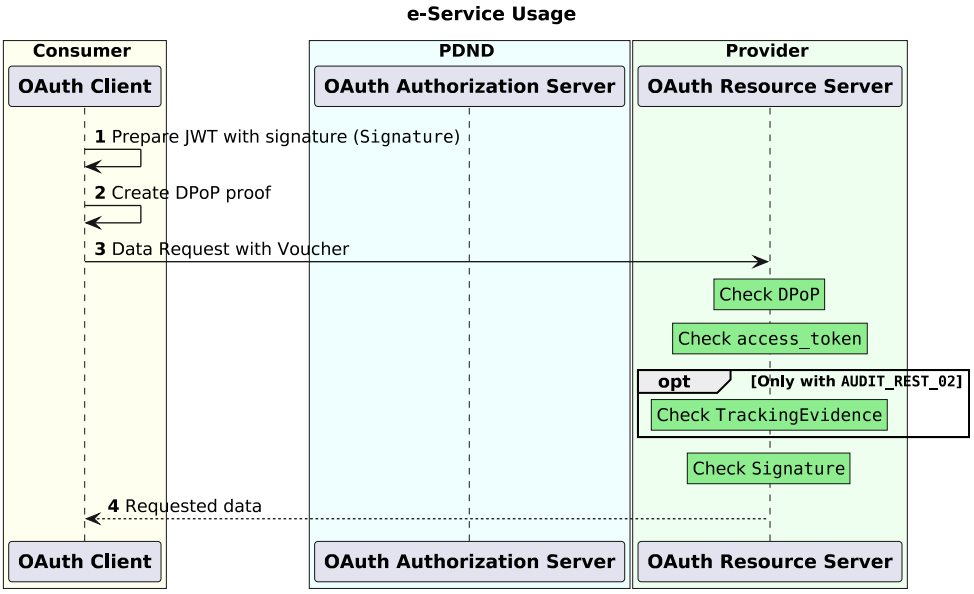


Fig. 20.6: Utilizzo dell'e-Service - Flusso dettagliato.


```
{
    "alg": "ES256",
    "kid": "d4c3b2a1-9876-5432-10fe-dcba98765432",
    "typ": "JWT"
}
```

```
{
    "iss": "9a8b7c6d-e5f4-g3h2-i1j0-klmnopqrstuv",
    "sub": "9a8b7c6d-e5f4-g3h2-i1j0-klmnopqrstuv",
    "aud": "https://erogatore.example/ente-example/v1",
    "iat": 1733397840,
    "nbf": 1733401628,
    "exp": 1733401440,
    "jti": "d3f7b2c9-274a-42b7-8f8d-2e9d8b1734b0",
    "signed_headers": [
        {"digest": "SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f
41"},
        {"content-type": "application/json"}
    ]
}
```


Nota

Il passo 1 è richiesto per rispettare il pattern di sicurezza `INTEGRITY_REST_02`.

Passo 2 ('DPoP proof'): Il Fruitore DEVE creare un nuovo JWT di *DPoP proof* seguendo le istruzioni fornite nella Sezione 4 di [**RFC 9449**] per la presentazione del

token all'Endpoint e-Service.

Nota

Il passo 2 è richiesto solo quando si rispetta l'implementazione `POP_DPoP`.

Passo 3 (Richiesta e-Service): Il Fruitore invia una *e-Service Request* all'Erogatore, includendo il Voucher.

```
POST /ente-example/v1/hello/echo/ HTTP/1.1
Host: erogatore.example
Authorization: DPoP eyJhbGciOiJFUzI1NiIsImtpZCI6ImI4MzlmNGM3LTFlNWQtNGE4YS05ZmM2LTcyZDNiN2
YwOTFlYyIsInR5cCI6ImF0K2p3dCJ9.eyJpc3MiOiJpbnRlcm9wLnBhZ29wYS5pdCIsInN1YiI6IjgyOTE0YjNmLTY
wYjItNDUyOS1iNGQ2LTNkNGU2N2YwYTkzMyIsImF1ZCI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1le
GFtcGxlL3YxIiwiZXhwIjoxNzMzMDQyMTUwLCJuYmYiOjE3MzMwNDE5NDUsImlhdCI6MTczMzA0MTkyMCwianRpIjo
iYzRmNWQ3ZTItYjdjOC00MGY2LTliNmEtZGM5YTRmNWFlYjU3IiwiY2xpZW50X2lkIjoiODI5MTRiM2YtNjBiMi00N
TI5LWI0ZDYtM2Q0ZTY3ZjBhOTMzIiwicHVycG9zZUlkIjoiZDJiOWE2NTMtYzQ5Ny00NWM2LWI4ZjEtNWJkZjEyNGM
5ZDNhIiwiZGlnZXN0Ijp7ImFsZyI6IlNIQTI1NiIsInZhbHVlIjoiOTkwOGQ5NGI4ZmViMjY4YzAzNzEwNmQ3Yzg5Z
TcwNjBjMmNjMWY2YjJiNGViY2I4MDViZmVlNTNhNTM5MzA3YiJ9LCJjbmYiOnsiamt0IjoiMFpjT0NPUlpOWXktRFd
wcXEzMGpaeUpHSFROMGQySGdsQlYzdWlndUE0SSJ9fQ.sGhaHEOfMTB7r4_8ZILM_a9eTBGawWn3kL-dxYoZggFIzy
rXDOZcQWt0zr00lMk2iYAMWxS32e4cUedmAsBXGw
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6IkVDIiwia2V5X29wcyI6WyJz
aWduIl0sImtpZCI6IjM5ZmE5NjBiLTc3M2YtNDllZi04YTBlLWU3NzNlOWI5N2FlOCIsImNydiI6IlAtMjU2IiwieC
I6Imh1eVhJUU52OTAyb0xzcFg0X3pvbkM5NEc2eUVsbjZsc2RtLTF3TTczMm8iLCJ5IjoiSTlQREVhd1dIcWFGREd4
MVprTmstMlBWNldkcGNhSDNBZk9iQlNMaWhndyJ9fQ.eyJqdGkiOiIyYzc2ZmNhMy1jYjRlLTQzMTItOGI2ZS05NzQ
5NDYyZjQyMGQiLCJodG0iOiJQT1NUIiwiYXRoIjoiM2UwOGRlMWQwYTNkZjIzNWZjZmNjZjYyNjdmYTUwYTU5YmEyY
Tk1NTI2YzdjZTY3MDY1YjhlMjZkYmI5NDQ1MSIsImh0dSI6Imh0dHBzOi8vZXJvZ2F0b3JlLmV4YW1wbGUvZW50ZS1
leGFtcGxlL3YxIiwiaWF0IjoxNzYyMjYyNjE2fQ.kvXh8H9B5DWCNlWyNB_PzRH217j1NHnIkE_55WnEixt2RbQTGr
CS6AFAznREA85dzqwAAaHb_qHtDc5BR0lLmQ
Agid-JWT-Signature: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5OD
c2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI5YThiN2M2ZC1lNWY0LWczaDItaTFqMC1rbG1ub3BxcnN0dXYiLCJ
zdWIiOiI5YThiN2M2ZC1lNWY0LWczaDItaTFqMC1rbG1ub3BxcnN0dXYiLCJhdWQiOiJodHRwczovL2Vyb2dhdG9yZ
S5leGFtcGxlL2VudGUtZXhhbXBsZS92MSIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE
3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZ
XJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDF
iZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.DpuBNo2U
gQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg
Digest: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f41
Agid-JWT-TrackingEvidence: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWR
jYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5
MzMiLCJhdWQiOiJodHRwczovL2Vyb2dhdG9yZS5leGFtcGxlL2VudGUtZXhhbXBsZS92MSIsImV4cCI6MTczMzA1Mj
YwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEy
LTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOG
k5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2
IiwibG9hIjoic3Vic3RhbnRpYWwifQ.bhb3f3aWEuK-bZWjyKRWrJ4hYUWhw2SQ-yRz0kUFjPQTVagjXuTqyhxsHO4
KXeSX9SivgaLSvw4n9BeZa7APbQ
```

```
Content-Type: application/json

{
    "parameter1": "value1",
    "parameter2": "value2"
}
```

L'Erogatore DEVE validare la *DPoP proof* [RFC 9449].

Nota

La validazione della *DPoP proof* è richiesta solo quando si rispetta l'implementazione `POP_DPoP`.

L'Erogatore DEVE validare il Voucher come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `at+jwt`.

Firma:

- Reperire l'insieme di chiavi pubbliche pubblicate all'Endpoint .well-known. Da questo insieme, selezionare la chiave pubblica il cui identificatore corrisponde al valore del parametro header `kid` nel Voucher.
- Validare la firma dell'`access_token` utilizzando la chiave pubblica recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- Il claim `iss` DEVE identificare il dominio dell'Authorization Server PDND.
- Il claim `sub` DEVE corrispondere al claim `client_id`.
- Il claim `aud` DEVE corrispondere all'e-Service previsto.
- In caso di implementazione `POP_DPoP`, il claim `cnf.jkt` DEVE corrispondere al SHA-256 Thumbprint della chiave pubblica DPoP nel claim `jwk` nella prova DPoP.

Nota

Se l'Erogatore richiede maggiori informazioni sul contesto della richiesta, può interagire con l'API di Interoperabilità PDND passando il valore del `purposeId` come parametro.

L'Erogatore DEVE validare il JWT `TrackingEvidence` come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `JWT`.

Firma:

- Ottenere la chiave pubblica del Fruitore corrispondente al parametro header `kid`, interagendo con l'API di Interoperabilità PDND.
- Validare la firma del JWT utilizzando la chiave pubblica del Fruitore recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- Il claim `iss` DEVE identificare il Client del Fruitore.
- Il claim `aud` DEVE identificare l'Erogatore.

Inoltre, l'Erogatore DEVE assicurarsi che l'hash del JWT `TrackingEvidence` corrisponda al valore del claim `digest.value` contenuto nel payload `access_token`.

Nota

La validazione del JWT `TrackingEvidence` è richiesta solo quando si rispetta il pattern di sicurezza `AUDIT_REST_02` o l'implementazione `POP_TPoP`.

L'Erogatore DEVE validare il JWT `Signature` come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `JWT`.

Firma:

- Validare la firma del JWT utilizzando la chiave pubblica del Fruitore recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- I claim `iss` e `sub` DEVONO identificare il Client del Fruitore.
- Il claim `aud` DEVE identificare l'Erogatore.

Inoltre, l'Erogatore DEVE validare l'integrità della *e-Service Request*, verificando che:

- Il claim `signed_headers.content-type` corrisponda al valore dell'header HTTP `Content-Type` della *e-Service Request*.

- Il claim `signed_headers.digest` corrisponda al valore del digest del payload della *e-Service Request*, nonché al valore dell'header HTTP `Digest` della *e-Service Request*.

Se uno qualsiasi dei controlli precedenti fallisce, l'Erogatore DEVE rifiutare la richiesta.

Passo 4 (Risposta): Qualora i controlli abbiano successo, l'Erogatore fornisce al Fruitore i dati richiesti. Nelle intestazioni della risposta DEVE essere presente il JWT (`Signature`) al fine di garantire l'integrità.

```
HTTP/1.1 200 OK
Content-Type: application/json
Agid-JWT-Signature: ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYj
BkMi03OWUzZjBjNGE4ZWYiLA0KICAidHlwIjogIkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3
OC1pOWowLWtsbW5vcHFyc3R3eCIsDQogICJzdWIiOiAiMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdH
d4IiwNCiAgImF1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6
IDE3MzMzOTc4NDAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZXhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6IC
I4ZTEyZjRiNy05YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAic2lnbmVkX2hlYWRlcnMiOiBbDQogICAg
ew0KICAgICAgImRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTgzMDYwMGFkOWJkY2E5OTM1OTNmOD
c2MjA5YTAwNGI1OTliNTgzMDk1YjBhNjEiDQogICAgfwNCiAgICB7DQogICAgICAiY29udGVudC10eXBlIjogImFw
cGxpY2F0aW9uL2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a
8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg
Digest: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b583095b0a61

{
  "requestedField1": "value1",
  "requestedField2": "value2",
  "requestedField3": "value3"
}
```

Il Fruitore DEVE validare il JWT `Signature` come segue:

Header:

- Assicurarsi che il claim `typ` sia presente e che il suo valore sia `JWT`.

Firma:

- Validare la firma del JWT utilizzando la chiave pubblica del Fruitore recuperata e l'algoritmo specificato dal parametro header `alg`.

Payload:

- I claim `iss` e `sub` DEVONO identificare l'Erogatore.
- Il claim `aud` DEVE identificare il Client del Fruitore.

Inoltre, il Fruitore DEVE validare l'integrità della *e-Service Response*, verificando che:

- Il claim `signed_headers.content-type` corrisponda al valore dell'header HTTP `Content-Type` della *e-Service Response*.
- Il claim `signed_headers.digest` corrisponda al valore del digest del payload della *e-Service Response*, nonché al valore dell'header HTTP `Digest` della *e-Service Response*.

Se uno qualsiasi dei controlli precedenti fallisce, il Fruitore DEVE rifiutare la richiesta.

Endpoint e-Service

Richiesta (e-Service)

La *e-Service Request* DEVE includere i seguenti parametri di header HTTP (se non diversamente specificato):

Parametro	Descrizione	Riferimento
Authorization	Voucher rilasciato dall'Authorization Server PDND.	[**RFC 9449**], [MODI], [PDND]
DPoP	JWT di *DPoP proof*, per rispettare il pattern di sicurezza `REST_JWS_2021_POP`. È obbligatorio solo quando si segue l'implementazione `POP_DPoP`.	[**RFC 9449**], [PDND]
Agid-JWT-Signature	JWT contenente la firma delle intestazioni del messaggio la cui integrità deve essere garantita, per rispettare il pattern di sicurezza `INTEGRITY_REST_02`.	[MODI]
Digest	Digest del payload del messaggio, per rispettare il pattern di sicurezza `INTEGRITY_REST_02`. Secondo **RFC 3230**, il formato DEVE essere il seguente: `<digest-algorithm>=<encoded digest output>`.	[**RFC 3230**], [MODI]
Agid-JWT-Tracking Evidence	JWT contenente i dati tracciati nel dominio del Fruitore. È obbligatorio solo quando si rispetta `AUDIT_REST_02` o l'implementazione `POP_TPoP`.	[MODI]

Il JWT `Signature`, contenuto nell'header HTTP `Agid-JWT-Signature`, DEVE includere i seguenti parametri nel JOSE header:

Parametro	Descrizione	Riferimento
alg	Identificativo di un algoritmo di firma digitale.	[**RFC 7515**]
kid	Identificativo univoco del JWK utilizzata dal Fruitore per firmare il JWT.	[**RFC 7515**]

continues on next page

Tabella 20.16 – continua dalla pagina precedente

Parametro	Descrizione	Riferimento
typ	DEVE essere impostato su `JWT`.	[**RFC 7515**], [**RFC 7519**]

Il JWT `Signature`, contenuto nell'header HTTP `Agid-JWT-Signature`, DEVE includere i seguenti claim nel payload:

Claim	Descrizione	Riferimento
iss	DEVE essere impostato sullo stesso valore di `client_id`.	[**RFC 7519**]
sub	DEVE essere impostato sullo stesso valore di `client_id`.	[**RFC 7519**]
aud	Identificativo dell'Erogatore.	[**RFC 7519**]
exp	Timestamp UNIX che rappresenta l'istante di scadenza del JWT.	[**RFC 7519**]
nbf	Timestamp UNIX che rappresenta il primo istante di validità del JWT (opzionale).	[**RFC 7519**]
iat	Timestamp UNIX che rappresenta l'istante di emissione del JWT.	[**RFC 7519**]
jti	Identificativo univoco del JWT per prevenire attacchi di replay.	[**RFC 7519**]
signed_headers	Oggetto JSON contenente le intestazioni firmate la cui integrità deve essere protetta, per rispettare `INTEGRITY_REST_02`. DEVE contenere i seguenti claim: • **digest**: stringa JSON che rappresenta la firma dell'header HTTP `Digest` • **content-type**: stringa JSON che rappresenta la firma dell'header HTTP `Content-Type`	[MODI]

Se presente, il JWT `TrackingEvidence`, contenuto nell'header HTTP `Agid-JWT-TrackingEvidence`, DEVE includere i seguenti parametri nel JOSE header:

Parametro	Descrizione	Riferimento
alg	Identificativo di un algoritmo di firma digitale.	[**RFC 7515**]
kid	Identificativo univoco del JWK utilizzato dal Fruitore per firmare il JWT.	[**RFC 7515**]
typ	DEVE essere impostato su `JWT`.	[**RFC 7515**], [**RFC 7519**]

Se presente, il JWT `TrackingEvidence`, contenuto nell'header HTTP `Agid-JWT-TrackingEvidence`, DEVE includere i seguenti claim nel payload:

Claim	Descrizione	Riferimento
iss	DEVE essere impostato sullo stesso valore di `client_id`.	[**RFC 7519**]
aud	Identificativo dell'Erogatore.	[**RFC 7519**]
exp	Timestamp UNIX che rappresenta l'istante di scadenza del JWT.	[**RFC 7519**]
nbf	Timestamp UNIX che rappresenta il primo istante di validità del JWT (opzionale).	[**RFC 7519**]
iat	Timestamp UNIX che rappresenta l'istante di emissione del JWT.	[**RFC 7519**]
jti	Identificativo univoco del JWT per prevenire attacchi di replay.	[**RFC 7519**]
purposeId	Identificativo dello scopo registrato nella Piattaforma PDND, associato all'e-Service previsto.	[MODI]
dnonce	DEVE essere una stringa casuale composta da numeri interi e con una lunghezza di 13 cifre.	[MODI]

Quando si rispetta il pattern di sicurezza `AUDIT_REST_02`, il payload `TrackingEvidence` DEVE contenere anche i dati tracciati concordati con l'Erogatore.

Risposta (e-Service)

La *e-Service Response* DEVE avere un content type `application/json`.

La *e-Service Response* DEVE includere i seguenti parametri di header HTTP:

Parametro	Descrizione	Riferimento
Agid-JWT-Signature	JWT contenente la firma delle intestazioni del messaggio la cui integrità deve essere garantita, per rispettare il pattern di sicurezza `INTEGRITY_REST_02`.	[MODI]
Digest	Digest del payload del messaggio, per rispettare il pattern di sicurezza `INTEGRITY_REST_02`. Secondo **RFC 3230**, il formato DEVE essere il seguente: `<digest-algorithm>=<encoded digest output>`.	[**RFC 3230**], [MODI]

Il JWT `Signature`, contenuto nell'header HTTP `Agid-JWT-Signature`, DEVE includere i seguenti claim nel payload:

Claim	Descrizione	Riferimento
iss	Identificativo dell'Erogatore.	[**RFC 7519**]
sub	Identificativo dell'Erogatore.	[**RFC 7519**]
aud	Identificativo del Fruitore.	[**RFC 7519**]
exp	Timestamp UNIX che rappresenta l'istante di scadenza del JWT.	[**RFC 7519**]
nbf	Timestamp UNIX che rappresenta il primo istante di validità del JWT (opzionale).	[**RFC 7519**]
iat	Timestamp UNIX che rappresenta l'istante di emissione del JWT.	[**RFC 7519**]
jti	Identificativo univoco del JWT per prevenire attacchi di replay.	[**RFC 7519**]
signed_headers	Oggetto JSON contenente le intestazioni firmate la cui integrità deve essere protetta, per rispettare `INTEGRITY_REST_02`. DEVE contenere i seguenti claim: • **digest**: stringa JSON che rappresenta la firma dell'header HTTP `Digest` • **content-type**: stringa JSON che rappresenta la firma dell'header HTTP `Content-Type`	[MODI]

Il payload del JWT della *e-Service Response* include specifici claim relativi ai dati forniti al Fruitore.

Se si verificano errori durante la validazione della *e-Service Request*, l'Endpoint e-Service DEVE restituire un errore, la cui struttura dipende dalla natura dell'errore.

In caso di problemi di autenticazione (cioè, Voucher non valido o scaduto), la risposta DEVE aderire al formato di errore definito in **RFC 6750#section-3** e **RFC 9449#section-7.1**, con specifico riferimento all'uso del parametro di header `WWW-Authenticate`.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: DPoP error="invalid_token", error_description="The access token expired"
```

Per tutti gli altri errori, la risposta DEVE aderire al formato di errore definito in **RFC 9457**. La risposta DEVE utilizzare `application/problem+json` (secondo MODI `RAC_GEN_FORMAT_002`) come `Content-Type` e DEVE includere i seguenti parametri:

- `type`: [OPZIONALE] URI che identifica il tipo di problema.
- `title`: [OBBLIGATORIO] Stringa contenente un breve riepilogo, leggibile dall'utente, del tipo di problema.
- `status`: [OBBLIGATORIO] Numero che indica il codice di stato HTTP relativo a questa specifica occorrenza del problema.

- `detail`: [OBBLIGATORIO] Stringa contenente una spiegazione, leggibile dall'utente, specifica per questa occorrenza del problema.
- `instance`: [OPZIONALE] Stringa contenente un riferimento URI che identifica la specifica occorrenza del problema.

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
    "title": "invalid_request",
    "status": 400,
    "detail": "The Agid-JWT-Signature header parameter is missing."
}
```

La seguente tabella elenca gli HTTP Status Code che DEVONO essere supportati per la risposta di errore:

HTTP Status Code	Descrizione
`400 Bad Request`	La richiesta non può essere soddisfatta perché mancano parametri richiesti, contiene parametri non validi o è in qualche modo malformata.
`401 Unauthorized`	La richiesta non può essere soddisfatta perché il Voucher è scaduto, revocato o in qualche modo malformato.
`500 Internal Server Error`	La richiesta non può essere soddisfatta perché l'Endpoint e-Service ha riscontrato un problema interno.
`503 Service Unavailable`	La richiesta non può essere soddisfatta perché l'Endpoint e-Service è temporaneamente non disponibile (ad esempio, a causa di manutenzione o sovraccarico).

20.3 PDND e-Service Template

La PDND fornisce uno strumento specializzato che migliora i processi di co-progettazione delle API ottimizzando la pubblicazione e il riutilizzo dei servizi elettronici. Questa funzionalità è definita e regolamentata nel presente documento.

- "Linee Guida sull'infrastruttura tecnologica della Piattaforma Digitale Nazionale Dati per l'interoperabilità dei sistemi informativi e delle basi di dati" (PDND).

Il template del servizio elettronico serve come modello standardizzato contenente tutti i metadati tecnici e descrittivi necessari per un servizio elettronico. I Gestori delle API, che possono essere sia Fornitori che Consumatori all'interno dell'ecosistema PDND, POSSONO creare e mantenere questi template.

Una volta che un e-service template è pubblicato, è accessibile attraverso il Catalogo Template PDND che è un repository centralizzato che facilita il riutilizzo. Questo catalogo consente a qualsiasi Partecipante PDND autorizzato di sfogliare i template disponibili e istanziare nuovi servizi elettronici basati su progetti esistenti.

20.3.1 Definizione e linee guida dell'e-service template PDND

L'infrastruttura PDND supporta la gestione del ciclo di vita dei Servizi Elettronici Template, simile a quella dei servizi elettronici tradizionali. Gli stati del ciclo di vita includono: **Draft**, **Active**, **Suspended** e **Deprecated**. Come per i servizi elettronici tradizionali, PDND applica il controllo degli accessi basato sui ruoli per governare le transizioni di stato.

Gestione dei Servizi Elettronici Template

Creazione del Servizio Elettronico Template

I Partecipanti sono abilitati a creare Servizi Elettronici Template tramite una procedura guidata accessibile attraverso l'interfaccia web PDND (le API saranno disponibili in futuro). Il flusso di lavoro di creazione rispecchia da vicino quello della creazione standard di servizi elettronici, con le seguenti distinzioni:

- Un campo aggiuntivo identifica il destinatario previsto del template.
- Il campo "Audience" è omesso.
- Le soglie sono opzionali e servono come raccomandazioni per i Partecipanti che implementano il template.

Ai Partecipanti è vietato creare più template con lo stesso nome: i nomi dei template DEVONO essere unici per partecipante. Alla creazione, un template è inizialmente impostato sullo stato Draft. I template possono quindi essere pubblicati nel Catalogo Template, rendendoli così accessibili a tutti i Partecipanti.

Modifica del Servizio Elettronico Template

I Partecipanti che hanno creato un template possono modificarlo. La portata dei campi modificabili dipende dallo stato del ciclo di vita del template:

- Se il template è in stato Draft, tutti i campi sono modificabili.
- Per i template in altri stati, solo un sottoinsieme limitato di campi può essere modificato direttamente.
- I campi che non possono essere modificati nei template pubblicati richiedono la creazione di una nuova versione del template per applicare le modifiche.

Il versionamento dei template funziona in modo simile a quello dei servizi elettronici, dato che le modifiche al modello possono impattare i servizi istanziati e quindi i Partecipanti che consumano quell'istanza.

I seguenti campi possono essere modificati senza attivare una nuova versione del template:

- Name
- Intended Recipient
- Description
- Voucher Time Limit
- Documentation (escludendo la specifica OpenAPI)
- Attributes

Sospensione del e-service Template

I template, come i servizi elettronici, possono essere Sospesi. Quando sospesi:

- Il template viene rimosso dal catalogo pubblico dei template.
- L'istanziazione di nuove istanze dal template sospeso è disabilitata.
- Le istanze precedentemente istanziate rimangono inalterate.
- I template possono essere riattivati in qualsiasi momento.
- I template non possono essere eliminati.

Istanziazione del e-service Template

I Partecipanti POSSONO istanziare un Servizio Elettronico Template sfogliando il Catalogo Template e selezionando un template. Questo processo genera un nuovo servizio elettronico.

I vincoli di istanziazione includono:

- Solo i template in stato Attivo sono idonei per l'istanziazione.
- L'istanziazione è facilitata attraverso una procedura guidata nell'interfaccia web PDND.
- A causa dell'obiettivo di standardizzazione dei template, la maggior parte dei campi è pre-compilata e immutabile durante l'istanziazione.
- Le seguenti informazioni non possono essere modificate durante l'istanziazione:
  - Caricamento della documentazione
  - Tempo di scadenza del token
  - Nome, descrizione e attributi

Invece, i seguenti campi devono essere specificati durante l'istanziazione:

- Audience
- Thresholds
- Automatic/Manual Approval Policy

Inoltre, sebbene la specifica OpenAPI sia fissa, i seguenti campi di metadati possono essere forniti in modo che PDND possa aggiornare automaticamente la specifica YAML:

- Contatti (nome, email, URL, URL Termini e Condizioni)
- URL del server

Ogni servizio elettronico istanziato mantiene un ciclo di vita indipendente analogo ai servizi elettronici standard.

Gestione delle Versioni

Il versioning dei template segue un processo controllato:

- La pubblicazione di una nuova versione del template la imposta su Active.
- La versione precedentemente Active viene automaticamente trasferita a Deprecated.
- È consentita solo una versione Active per template in qualsiasi momento.
- I template possono anche avere una singola versione Draft che coesiste con la versione Active.

Le istanze derivate dai template mantengono un versioning indipendente poiché i Partecipanti possono aggiornare i campi specifici dell'istanza (ad esempio, URL del server) più volte, mentre l'istanza rimane collegata alla versione del template di origine.

Di conseguenza, le versioni dei template e le versioni delle istanze sono indipendenti e non direttamente correlate.

I Partecipanti che istanziano un template possono quindi aggiornare sia l'istanza specifica o, se disponibile, aggiornare a una versione più recente del template.

Template per Fonte Autentica

La funzionalità del servizio elettronico template viene utilizzata per standardizzare la trasmissione dei dati dalle Fonti Autentiche ai Fornitori di Attestati Elettronici. Il servizio elettronico template DOVREBBE essere pubblicato all'interno della PDND dal Fornitore di Attestati Elettronici ed è accessibile attraverso il Catalogo Template PDND.

Parametri del Template per Fonte Autentica

Il servizio elettronico template **DEVE** rispettare le seguenti proprietà:

- **Name**: Creazione EAA <`Nome / Nome tipologia EAA`> – IT-Wallet
- **Intended Recipients**: IT Wallet - Fonte Autentica - <`Dominio della Fonte Autentica`>
- **Description**: Descrizioni utili al Fornitore di Attestati Elettronici in relazione al nuovo attestato elettronico <`Nome dell'Attestato Elettronico`>
- **Technology**: REST
- **Data variation via Signal Hub**: True

- **Version changelog**: Servizio elettronico Fonte Autentica tramite implementazione template
- **Voucher Time Limit**: 20
- **Suggest custom threshold**: False
- **Suggest manual agreement approval policy**: False
- **Attributes**: `<Nome ufficiale dell'Ente Pubblico Fornitore di Attestati Elettronici>`

Istanziazione del Template per Fonte Autentica

Ogni Fonte Autentica **DOVREBBE** istanziare il servizio elettronico template *IT Wallet - Fonte Autentica* nella PDND. Il processo di istanziazione risulterà in un nuovo servizio elettronico che **DEVE** soddisfare i seguenti requisiti:

- **Signal Hub**: True
- **Politica di approvazione manuale**: False
- **Soglia giornaliera chiamate API per ogni fornitore**: maggiore di 10000
- **Soglia giornaliera chiamate API**: maggiore di 10000

Informazioni aggiuntive richieste durante il processo di creazione sono dipendenti dal fornitore.

Specifica OpenAPI della Fonte Autentica PDND

Di seguito è riportata la specifica OpenAPI completa per i servizi elettronici della Fonte Autentica PDND:

```yaml
openapi: 3.0.1
info:
  title: IT Wallet API - AS web services
  version: 0.2.0
  description: |
    # IT Wallet Authentic Source e-Service exposed via PDND.
    ### ModI patterns to be used:
      - <b>ID_AUTH_CHANNEL_01</b>: Direct Trust TLS (HTTPS)
      - <b>ID_AUTH_REST_01</b>: Authorization via PDND token (always required)
      - <b>INTEGRITY_REST_02</b>: Requests and responses are signed
      - <b>REST_JWS_2021_POP</b>: Proof of possession of the PDND Voucher (POP_DPoP per RFC 9449, or POP_TPoP)
```

```
12        - <b>AUDIT_REST_02</b>: Optional. The Provider MAY require additional tracked data f
    rom the Consumer; if so, this pattern MUST be used, including together with POP_DPoP, via
    the optional `Agid-JWT-TrackingEvidence` header
13    termsOfService: "https://authentic-source.example.it/tos/"
14    contact:
15      name: IT-Wallet <credential_name> <credential_provider>
16      url: https://github.com/italia/eid-wallet-it-docs
17    x-api-id: ASITW-01
18    x-summary: IT-Wallet Authentic Source API.
19  servers:
20    - url: https://test.authentic-source.example.it/v0.2.0
21      description: Authentic Source API test server
22    - url: https://authentic-source.example.it/v0.2.0
23      description: Authentic Source API production server
24  security:
25    - DPoPVoucherAuth: []
26      DPoPProof: []
27    - BearerAuth: []
28      TrackingEvidenceAuth: []
29  paths:
30    /status:
31      get:
32        tags:
33          - status
34        summary: Get Authentic Source API status.
35        description: Health-check endpoint that returns the operational status of the Authen
    tic Source API.
36        operationId: authenticSourceStatus
37        responses:
38          "200":
39            description: Service available
40            content:
41              application/problem+json:
42                schema:
43                  $ref: "#/components/schemas/ProblemDetails"
44            headers:
45              Cache-Control:
46                $ref: "#/components/headers/CacheControlHeader"
47              X-RateLimit-Limit:
48                $ref: "#/components/headers/RateLimitLimitHeader"
49              X-RateLimit-Remaining:
50                $ref: "#/components/headers/RateLimitRemainingHeader"
51              X-RateLimit-Reset:
52                $ref: "#/components/headers/RateLimitResetHeader"
53          "429":
54            description: Too Many Requests
55            content:
```

```
56              application/problem+json:
57                schema:
58                  $ref: "#/components/schemas/ProblemDetails"
59            headers:
60              # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X
    -RateLimit-Reset are not required because redundant along with Retry-After.
61              Retry-After:
62                $ref: "#/components/headers/RetryAfterHeader"
63          "503":
64            description: Service Unavailable
65            content:
66              application/problem+json:
67                schema:
68                  $ref: "#/components/schemas/ProblemDetails"
69            headers:
70              Retry-After:
71                $ref: "#/components/headers/RetryAfterHeader"
72
73    /attribute-claims/{datasetId}:
74      post:
75        tags:
76          - credential
77        summary: Get Attribute Claims
78        description: >-
79          This service provides the Credential Issuer with all attribute claims necessary fo
    r the issuance of a Digital Credential
80        operationId: attributeClaims
81        parameters:
82          - in: path
83            name: datasetId
84            schema:
85              type: string
86            required: true
87            description: Identifier of the dataset as registered in the Authentic Source Reg
    istry
88          - name: Agid-JWT-Signature
89            in: header
90            description: >-
91              JWT containing the signature of the message headers whose integrity
92              needs to be guaranteed, to comply with the INTEGRITY_REST_02
93              security pattern (see <a target="blank"
94              href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-servi
    ce-pdnd.html">e-Service PDND</a>). <br/><br/>
95
```

```
96              <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA">EXAMPLE
97              ON JWT.IO</a>
98            required: true
99            schema:
100             type: string
101             format: JWT
102             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA
103         - name: Digest
104           in: header
105           description: >-
106             Digest of the message payload, to comply with the INTEGRITY_REST_02
107             security pattern. According to <a target="blank" href="https://www.rfc-editor.org/rfc/rfc3230.html#section-4.2">RFC
108             3230 §4.2</a>, the format MUST be the following: digest-algorithm=encoded
109             digest output.
110           required: true
111           schema:
112             type: string
113             example: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f41
114         - name: Agid-JWT-TrackingEvidence
115           in: header
116           description: >-
117             TrackingEvidence JWT for AUDIT_REST_02. Optional. It MAY be sent
118             together with POP_DPoP (`Authorization: DPoP` and the `DPoP` proof)
119             when the Provider requires tracked data from the Consumer, such as
120             caller LoA, userID, IP address, or other data agreed with the
121             Provider. The DPoP proof remains the proof of possession. When the
122             Provider mandates AUDIT_REST_02, this header MUST be present.
123             For POP_TPoP the same header is required by `TrackingEvidenceAuth`
124             together with `BearerAuth`; send a single header value
125             (see <a target="blank"
```

```
126             href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pdnd.html">e-Service PDND</a>). <br/><br/>
127             <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA">EXAMPLE
128             ON JWT.IO</a>
129           required: false
130           schema:
131             type: string
132             format: JWT
133             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA
134       requestBody:
135         required: true
136         content:
137           application/json:
138             schema:
139               $ref: "#/components/schemas/CredentialClaimsRequest"
140       responses:
141         "200":
142           description: OK
143           headers:
144             Agid-JWT-Signature:
```

```
145              description: JWT containing the signature of the message headers whose integ
    rity needs to be guaranteed, to comply with the INTEGRITY_REST_02 security pattern (see <a
     target="blank" href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-s
    ervice-pdnd.html">e-Service PDND</a>). <a target="blank" href="https://jwt.io/#debugger-io
    ?token=ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYjBkMi03OWUzZjB
    jNGE4ZWYiLA0KICAidHlwIjogIkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3OC1pOWowLWtsb
    W5vcHFyc3R3eCIsDQogICJzdWIiOiAiMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdHd4IiwNCiAgImF
    1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6IDE3MzMzOTc4N
    DAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZXhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6ICI4ZTEyZjRiNy0
    5YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAic2lnbmVkX2hlYWRlcnMiOiBbDQogICAgew0KICAgICAgI
    mRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTgzMDYwMGFkOWJkY2E5OTM1OTNmODc2MjA5YTAwNGI
    1OTliNTgzMDk1YjBhNjEiDQogICAgfSwNCiAgICB7DQogICAgICAiY29udGVudC10eXBlIjogImFwcGxpY2F0aW9uL
    2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPvCX6HfktkxG7L5mk6a8OK1Hg0mQcZfF
    i3gelS-aL9kFS-6MoSy4csBg">EXAMPLE
146              required: true
147              schema:
148                type: string
149            Digest:
150              description: Digest of the message payload, to comply with the INTEGRITY_RES
    T_02 security pattern. According to RFC 3230 Section 4.2 <a target="blank" href="https://w
    ww.rfc-editor.org/rfc/rfc3230.html#section-4.2">RFC 3230 §4.2</a>, the format MUST be the
    following digest-algorithm=encoded digest output.
151              required: true
152              schema:
153                type: string
154                example: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b5830
    95b0a61
155            Cache-Control:
156              $ref: "#/components/headers/CacheControlHeader"
157            X-RateLimit-Limit:
158              $ref: "#/components/headers/RateLimitLimitHeader"
159            X-RateLimit-Remaining:
160              $ref: "#/components/headers/RateLimitRemainingHeader"
161            X-RateLimit-Reset:
162              $ref: "#/components/headers/RateLimitResetHeader"
163          content:
164            application/json:
165              schema:
166                $ref: "#/components/schemas/CredentialClaimsResponse"
167              example:
168                interval: 864000
169                userClaims:
170                  given_name: "Mario"
171                  family_name: "Rossi"
172                  birth_date: "1980-01-10"
173                  place_of_birth:
174                    country: "IT"
```

```
                  region: "Roma"
                  locality: "Roma"
                tax_id_code: "TINIT-RSSMRA80A01H501Z"
                personal_administrative_number: "12345A123A"
              attributeClaims:
                - object_id: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
                  status: "VALID"
                  last_updated: "2025-01-15T10:30:00Z"
                  institute_name: "Nome Istituto Universitario"
                  programme_type_name: "Laurea Magistrale"
                  degree_course_name: "Computer Science - Informatica"
                  academic_qualification_date: "2025-06-25"
                - object_id: "7A0720AB-9C97-E122-C53E-11D05FD075GG"
                  status: "VALID"
                  last_updated: "2025-01-10T08:00:00Z"
                  institute_name: "Nome Istituto Universitario"
                  programme_type_name: "Laurea Triennale"
                  degree_course_name: "Informatica"
                  academic_qualification_date: "2022-11-27"
              metadataClaims:
                - object_id: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
                  issuance_date: "2025-06-25"
                - object_id: "7A0720AB-9C97-E122-C53E-11D05FD075GG"
                  issuance_date: "2022-11-27"
        "400":
          description: Bad Request
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            X-RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "401":
          description: Unauthorized
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            X-RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            X-RateLimit-Remaining:
```

```
222                $ref: "#/components/headers/RateLimitRemainingHeader"
223              X-RateLimit-Reset:
224                $ref: "#/components/headers/RateLimitResetHeader"
225              WWW-Authenticate:
226                $ref: "#/components/headers/WWWAuthenticateHeader"
227          "404":
228            description: Claims not found
229            content:
230              application/problem+json:
231                schema:
232                  $ref: "#/components/schemas/ProblemDetails"
233            headers:
234              X-RateLimit-Limit:
235                $ref: "#/components/headers/RateLimitLimitHeader"
236              X-RateLimit-Remaining:
237                $ref: "#/components/headers/RateLimitRemainingHeader"
238              X-RateLimit-Reset:
239                $ref: "#/components/headers/RateLimitResetHeader"
240          "429":
241            description: Too Many Requests
242            content:
243              application/problem+json:
244                schema:
245                  $ref: "#/components/schemas/ProblemDetails"
246            headers:
247              # RFC 6585 defines Retry-After. X-RateLimit-Limit, X-RateLimit-Remaining and X
    -RateLimit-Reset are not required because redundant along with Retry-After.
248              Retry-After:
249                $ref: "#/components/headers/RetryAfterHeader"
250          "500":
251            description: Internal Server Error.
252            content:
253              application/problem+json:
254                schema:
255                  $ref: "#/components/schemas/ProblemDetails"
256            headers:
257              Retry-After:
258                $ref: "#/components/headers/RetryAfterHeader"
259          "503":
260            description: Service Unavailable
261            content:
262              application/problem+json:
263                schema:
264                  $ref: "#/components/schemas/ProblemDetails"
265            headers:
266              Retry-After:
267                $ref: "#/components/headers/RetryAfterHeader"
```

```yaml

tags:
  - name: status
    description: Endpoint di health check dell'API.
  - name: credential
    description: Retrieve information about the credential.

components:
  securitySchemes:
    DPoPVoucherAuth:
      type: http
      scheme: DPoP
      description: >-
        PDND Voucher issued with DPoP binding (POP_DPoP, RFC 9449). The
        `Authorization` header MUST be `DPoP <voucher>` (`token_type` is `DPoP`).
        This scheme MUST be used together with `DPoPProof`. It MUST NOT be
        modelled as HTTP Bearer: a conformant Provider rejects
        `Authorization: Bearer` for a DPoP-bound Voucher.
    DPoPProof:
      type: apiKey
      in: header
      name: DPoP
      description: >-
        DPoP proof JWT (RFC 9449). REQUIRED together with `DPoPVoucherAuth`
        (POP_DPoP). MUST NOT be sent with `BearerAuth` (POP_TPoP), where proof
        of possession is `Agid-JWT-TrackingEvidence`.
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: >-
        PDND Voucher issued as Bearer (POP_TPoP). The `Authorization` header
        MUST be `Bearer <voucher>` (`token_type` is `Bearer`). This scheme MUST
        be used together with `TrackingEvidenceAuth` as proof of possession.
        This is REST_JWS_2021_POP via TrackingEvidence, not REST_JWS_2021_Bearer.
    TrackingEvidenceAuth:
      type: apiKey
      in: header
      name: Agid-JWT-TrackingEvidence
      description: >-
        TrackingEvidence JWT. REQUIRED together with `BearerAuth` (POP_TPoP) as
        proof of possession of the PDND Voucher. The same header is also an
        optional operation parameter, so it MAY be sent with the POP_DPoP pair
        (`DPoPVoucherAuth` + `DPoPProof`) when the Provider requires
        AUDIT_REST_02. The DPoP proof remains the proof of possession.
        See e-Service PDND in the Technical Specifications.

```

```
315     headers:
316       CacheControlHeader:
317         schema:
318           type: string
319           enum:
320             - no-store
321         description: no-store
322       RateLimitLimitHeader:
323         schema:
324           type: integer
325           format: int32
326           minimum: 0
327         description: Maximum number of requests within the time window.
328       RateLimitRemainingHeader:
329         schema:
330           type: integer
331           format: int32
332           minimum: 0
333         description: Remaining requests within the time window.
334       RateLimitResetHeader:
335         schema:
336           type: integer
337           format: int32
338           minimum: 0
339         description: UTC epoch in seconds, corresponding to when the window for the current
    rate limit will reset.
340       RetryAfterHeader:
341         schema:
342           type: integer
343           format: int32
344           minimum: 0
345         description: Seconds to wait before receiving another response.
346       WWWAuthenticateHeader:
347         schema:
348           type: string
349           example: >-
350             Bearer error="invalid_token", error_description="The access token expired"
351         description: The request cannot be fulfilled because the Voucher is expired, revoked
     or otherwise malformed. See <a target="blank" href="https://datatracker.ietf.org/doc/html
    /rfc6750.html#section-3">RFC6750</a> and <a target="blank" href="https://datatracker.ietf.
    org/doc/html/rfc9449.html#section-7.1-11">RFC9449</a> for details.
352
353     schemas:
354       CredentialClaimsResponse:
355         type: object
356         properties:
357           userClaims:
```

```
358            description: List of User Claims.
359            type: object
360            properties:
361              given_name:
362                description: Current First Name.
363                type: string
364                example: "Mario"
365              family_name:
366                description: Current Family Name.
367                type: string
368                example: "Rossi"
369              birth_date:
370                description: Date of Birth.
371                type: string
372                example: "1980-01-10"
373              place_of_birth:
374                $ref: "#/components/schemas/place_of_birth"
375              tax_id_code:
376                description: National tax identification number. REQUIRED if personal_admini
    strative_number is absent.
377                type: string
378                example: "TINIT-XXXXXXXXXXXXXXX"
379              personal_administrative_number:
380                description: National unique identifier of a natural person. REQUIRED if tax
    _id_code is absent.
381                type: string
382                example: "XX00000XX"
383          attributeClaims:
384            description: List of Datasets of Attribute.
385            type: array
386            items:
387              type: object
388              properties:
389                object_id:
390                  description: Unique identifier of the Dataset. It MUST NOT contain persona
    l data. Required also if additionalProperties are not present. This parameter MUST be used
     to notify through Signal Hub data updating or availability.
391                  type: string
392                  example: "6F9619FF-8B86-D011-B42D-00C04FC964FF"
393                issuance_date:
394                  description: Administrative validity start date of the Dataset
395                  type: string
396                  example: "2025-01-01"
397                expiry_date:
398                  description: Administrative expiry date of the Dataset.
399                  type: string
400                  example: "2025-12-31"
```

```
401             additionalProperties:
402               oneOf:
403                 - type: string
404                 - type: integer
405                   format: int32
406                 - type: integer
407                   format: int64
408                 - type: number
409                   format: float
410                 - type: number
411                   format: double
412                 - type: boolean
413                 - type: array
414                   items:
415                     $ref: '#/components/schemas/additionalPropertiesArray'
416                 - type: object
417                   additionalProperties:
418                     $ref: '#/components/schemas/additionalPropertiesObject'
419             required: [object_id]
420         metadataClaims:
421           description: List of Metadata of Attribute.
422           type: array
423           items:
424             type: object
425             properties:
426               object_id:
427                 description: Unique identifier of the Dataset. It MUST NOT contain persona
    l data.
428                 type: string
429                 example: ”6F9619FF-8B86-D011-B42D-00C04FC964FF”
430               description:
431                 description: Human-Readable description of the Dataset.
432                 type: string
433                 example: ”Example: Master's Degree in Computer Science”
434               status:
435                 description: |
436                   Stato del Dataset. OBBLIGATORIO se additionalProperties di attributeClai
    ms sono presenti. I dataset Issued e Expired ricadono in VALID; la scadenza è verificata
437                   tramite metadata claims (es. expiry_date, nbf/exp). INVALID indica revoc
    a attiva da parte dell'AS.
438                   Per come questo stato influenza il ciclo di vita dell'Attestato Elettron
    ico gestito dal
439                   Fornitore di Attestati Elettronici, vedere la sezione <a target=”blank”
    href=”https://italia.github.io/eid-wallet-it-docs/versione-corrente/it/credential-revocati
    on.html#aggiornamento-dello-stato-da-parte-delle-fonti-autentiche”>Aggiornamento dello Sta
    to da parte delle Fonti Autentiche</a>.
440                 type: string
```

```
441                 enum:
442                   - VALID
443                   - INVALID
444                   - SUSPENDED
445                 x-enum-description:
446                   - VALID - Il dataset è valido (include Issued e Expired; la scadenza è v
    erificata via metadata).
447                   - INVALID - Il dataset è stato attivamente revocato dalla Fonte Autentic
    a.
448                   - SUSPENDED - Il dataset è temporaneamente non valido (tipicamente rever
    sibile).
449                 example: "VALID"
450               status_description:
451                 description: Human-Readable description of the Status.
452                 type: string
453                 example: "Example: Master's Degree in Computer Science"
454               last_updated:
455                 description: OBBLIGATORIO se additionalProperties di attributeClaims sono
    presenti. Last time the status or attributes of the Dataset have been updated. Its format
    is `YYYY-MM-DDTHH:MM:SSZ`.
456                 type: string
457                 example: "2025-01-15T10:30:00Z"
458               interval:
459                 description: Required if userClaims and additionalProperties of AttributeC
    laim parameters are not present. This represents the estimated amount of time (in seconds)
     required before making the request of the attribute claims again.
460                 type: integer
461                 format: int64
462                 example: 864000
463             required: [object_id, description]
464       required: [attributeClaims, metadataClaims]
465     CredentialClaimsRequest:
466       required:
467         - unique_id
468       type: object
469       properties:
470         unique_id:
471           type: string
472           description: ID ANPR or Tax identification number
473         object_id:
474           type: string
475           description: Unique identifier of the Credential dataset. If this parameter is p
    resent only the indicated dataset is returned.
476     additionalPropertiesArray:
477       type: array
478       items:
479         oneOf:
```

```yaml
        - type: string
        - type: integer
          format: int32
        - type: integer
          format: int64
        - type: number
          format: float
        - type: number
          format: double
        - type: boolean
        - type: object
          additionalProperties:
            $ref: '#/components/schemas/additionalPropertiesObject'
    additionalPropertiesObject:
      type: object
      oneOf:
        - type: string
        - type: integer
          format: int32
        - type: integer
          format: int64
        - type: number
          format: float
        - type: number
          format: double
        - type: boolean
    place_of_birth:
      description: Place of Birth.
      type: object
      properties:
        country:
          description: "alpha-2 country code as specified in ISO 3166-1"
          type: string
          example: "IT"
        region:
          description: "Name of a state, province, district, or local area"
          type: string
          example: "Roma"
        locality:
          description: "Name of a municipality, city, town, or village"
          type: string
          example: "Roma"
      anyOf:
        - required: [country]
        - required: [region]
        - required: [locality]
    ProblemDetails:
```

```
527       type: object
528       description: RFC7807-compliant problem details object for error responses.
529       properties:
530         type:
531           type: string
532           format: uri
533           description: An absolute URI that identifies the problem type.
534         title:
535           type: string
536           description: A short, human-readable summary of the problem type.
537         status:
538           type: integer
539           format: int32
540           description: The HTTP status code generated by the origin server for this occurr
    ence of the problem.
541         detail:
542           type: string
543           description: A human-readable explanation specific to this occurrence of the pro
    blem.
544         instance:
545           type: string
546           format: uri
547           description: An absolute URI that identifies the specific occurrence of the prob
    lem.
548       required: [title, status, detail]
```

20.4 Test Plans

Lo scopo dei piani di test è supportare implementatori, auditor e ambienti di test di conformità nella validazione del comportamento delle Soluzioni Wallet, delle Relying Party e degli Provider di Credenziali in vari scenari operativi e di sicurezza.

Tutti i casi di test sono derivati da regole normative definite nelle specifiche sopra indicate, senza presupposti o estensioni.

Nota

Si prega di notare che la matrice dei piani di test potrebbe essere soggetta a modifiche future.

20.4.1 Struttura della Matrice di Test

Ogni test è identificato da un identificatore univoco (es. WS-001) e categorizzato utilizzando i domini funzionali definiti di seguito.

- **Discovery**
- **Security**
- **Privacy**
- **Authorization**
- **Authentication**
- **User Experience**
- **Credential Issuance**
- **Credential Presentation**
- **Credential Status**
- **Backup & Restore**

Per ogni caso di test, la tabella specifica:

- **Test Case ID**: un identificatore univoco.
- **Purpose**: l'area funzionale coperta dal test.
- **Description**: il requisito che viene testato, sempre basato su un MUST normativo della specifica.
- **Expected Result**: il risultato atteso quando la soluzione è implementata correttamente.

Matrice di Test per la Valutazione delle Firme

Questa sezione fornisce l'insieme comune di casi di test per le Soluzioni Wallet, le Relying Party e i Credential Issuer per la valutazione di qualsiasi dichiarazione firmata, siano queste asserzioni, richieste, attestazioni o Credenziali.

Test Case ID	Purpose	Description	Expected Result
ATT-001	Discovery, Security	Valutazione dell'emittente	Le entità che valutano dichiarazioni firmate stabiliscono fiducia con l'emittente e ne valutano la conformità. Emittenti non scopribili all'interno della federazione o non collegabili a qualsiasi Trust Anchor noto, interrompono qualsiasi comunicazione di protocollo.
ATT-002	Discovery, Security	Valutazione della firma	Le entità valutano le dichiarazioni firmate verificando la firma con il materiale crittografico dell'emittente, a condizione che sia considerato attendibile attraverso un Trust Anchor ben noto. Qualsiasi materiale crittografico non attendibile o firme non valide interrompono le comunicazioni di protocollo.
ATT-003	Algorithm Verification	Verificare che l'algoritmo specificato nell'header corrisponda a quello utilizzato per le operazioni crittografiche.	L'algoritmo nell'header deve corrispondere all'operazione crittografica.
ATT-004	Appropriate Algorithms	Assicurarsi che siano usati solo gli algoritmi elencati come MUST o RECOMMENDED in *Algoritmi Crittografici* (pagina 647). Gli algoritmi elencati come MUST NOT (incluso `none`) sono rifiutati. Cipher suite e funzioni di hash non profilate in quella sezione seguono le Linee guida funzioni crittografiche ACN. Curve di Edwards e algoritmi post-quantum non sono obbligatori nel profilo attuale.	Solo gli algoritmi approvati sono accettati; quelli deprecati o non elencati sono rifiutati.
ATT-005	Signature Validation	Validare tutte le operazioni crittografiche e rifiutare se qualsiasi operazione fallisce.	Tutte le firme devono essere valide; qualsiasi fallimento risulta in un rifiuto.

continues on next page

Tabella 20.23 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
ATT-006	Key Entropy	Chiavi crittografiche e segreti freschi forniscono almeno 128 bit di security strength come definito in NIST SP 800-57 Part 1. Le chiavi asimmetriche usate con ES256/ESP256 sono chiavi P-256 generate nel CSPRNG del Keystore o WSCD. Nonce, `jti`, `state` e valori analoghi sono output CSPRNG di almeno 128 bit e non sono sequenziali.	Chiavi e segreti soddisfano i minimi di entropia; chiavi deboli, sottodimensionate, sequenziali o forgiate in software sono rifiutate. Una terza parte verifica tipo e lunghezza della chiave e, ove la piattaforma lo esponga, che la chiave sia in hardware sicuro (Android `KeyInfo.isInsideSecureHardware` / token Secure Enclave iOS).
ATT-006a	Key Entropy	PKCE `code_verifier`	Il `code_verifier` ha da 43 a 128 caratteri unreserved come richiesto da **RFC 7636**, generato con un CSPRNG.
ATT-006b	Key Entropy	Nonce MRTD e challenge	Gli identificativi di challenge e i nonce MRTD PoP hanno almeno 128 bit di entropia, come richiesto per l'Autenticazione eID Substantial con Verifica MRTD.
ATT-006c	Key Entropy	Nessuna chiave debole importata	Il Fornitore del Wallet rifiuta Key Attestation per chiavi importate nel Keystore, generate fuori dal Keystore/ WSCD, o la cui lunghezza non corrisponde all'algoritmo (ad esempio ES256 con una chiave non P-256).
ATT-007	Issuer Validation	Validare che le chiavi crittografiche appartengano all'emittente.	Le chiavi devono essere verificate come appartenenti all'emittente.
ATT-008	Audience Validation	Validare il claim audience per assicurarsi che il token sia utilizzato dalla parte destinata.	Il claim audience deve corrispondere al destinatario previsto.
ATT-009	Claim Trust	Non fidarsi dei claim ricevuti senza validazione.	I claim devono essere validati; i claim non attendibili sono rifiutati.
ATT-010	Explicit Typing	Utilizzare il typing esplicito per prevenire confusione COSE/JOSE.	Il typing deve essere esplicito e validato.

continues on next page

Tabella 20.23 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
ATT-011	Cross-JWT Confusion	Prevenire l'uso di COSE/ JOSE in contesti non intesi.	COSE/JOSE deve essere validato contestualmente per prevenire uso improprio.
ATT-012	Substitution Attacks	Assicurarsi che COSE/JOSE non siano sostituiti in contesti diversi.	COSE/JOSE deve essere validato per l'uso specifico del contesto.
ATT-013	Issued At Validation	Verificare che il parametro *issued at* sia impostato all'ora corrente, permettendo un periodo di grazia non superiore a 120 secondi.	Il valore *issued at* deve essere entro 120 secondi dall'ora corrente.
ATT-014	Expiration Validation	Assicurarsi che il tempo di *expiration* sia maggiore del tempo *issued at*.	Il tempo di *expiration* deve essere successivo al tempo *issued at*.
ATT-015	Data model validation	Assicurarsi che il tipo JOSE/ COSE corrisponda al data model definito.	I parametri o claim, i loro valori e lo schema utilizzato per rappresentarli sono conformi al data model.

Matrice di Test per la Valutazione della Fiducia

Questa sezione fornisce l'insieme comune di casi di test per le Soluzioni Wallet, le Relying Party e i Credential Issuer.

Test Case ID	Purpose	Description	Expected Result
ALL-001	Security	Ottenimento dei materiali crittografici pubblici dei Trust Anchors	Le entità ottengono la lista dei Trust Anchors o Certificate Authorities e i loro materiali di chiavi crittografiche pubbliche, assicurandosi periodicamente che questi non siano scaduti, revocati o aggiornati. L'infrastruttura di fiducia fornisce queste informazioni attraverso web endpoint e altri meccanismi out of band, per facilitare il confronto delle informazioni fornite a tutte le entità.

continues on next page

Tabella 20.24 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
ALL-002	Security	Auto-valutazione della conformità	Le entità valutano periodicamente la loro conformità e presenza all'interno della federazione, controllando che la trust chain su se stesse sia ancora valida, non revocata e conforme alla specifica tecnica. Le entità applicano le policy, controllando che la loro configurazione attuale sia valida con le policy attive su di loro all'interno della federazione. Le trust chain, valutate e memorizzate in formati multipli per facilitare l'interoperabilità nella discovery della fiducia con altre entità, sono memorizzate dalle entità e utilizzate all'occorrenza durante i flussi di scambio dati. Le trust chain sulle entità vengono recuperate o scoperte utilizzando le asserzioni emesse dalle entità.
ALL-003	Discovery	Pubblicazione delle informazioni su se stessi	Le entità firmano e pubblicano tutte le informazioni su di loro, contenenti tutti i metadata del protocollo, materiale crittografico, trust marks, utilizzando il well-known endpoint definito in questa specifica, rendendo queste informazioni pubblicamente scopribili da altre entità.

continues on next page

Tabella 20.24 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
ALL-004	Security	Pubblicazione del registro storico delle chiavi	Le entità firmano e pubblicano tutte le informazioni sui materiali crittografici non utilizzati o revocati utilizzando well-known endpoint definiti in questa specifica, rendendo queste informazioni pubblicamente scopribili da altre entità.
ALL-005	Security	Valutazione della conformità con le entità prima dello scambio di dati sull'Utente	Le entità valutano la fiducia e la conformità con altre entità prima che qualsiasi informazione relativa a una persona fisica o giuridica possa essere scambiata. Configurazioni false non permettono scambi di dati.
ALL-006	Security	Valutazione della prova di possesso durante l'uso di un'asserzione firmata in accordo con il metodo di conferma della proprietà di utilizzo configurato.	Le entità valutano il metodo di conferma e applicano il suo protocollo per considerare valida la dichiarazione firmata.
ALL-007	Security	Algoritmi crittografici supportati	Le entità valutano la crittografia utilizzata in conformità con gli algoritmi consentiti.
ALL-008	Security	Attacchi replay	Le dichiarazioni firmate utilizzando identificatori univoci sono memorizzate fino al loro tempo di scadenza e controllate contro qualsiasi replay di esse.

Matrice di Test per le Autorità di Federazione

Questa sezione definisce i casi di test per le Autorità di Federazione (Trust Anchor e Intermediari) responsabili del funzionamento dell'Infrastruttura di Trust come descritto in *L'Infrastruttura di Trust* (pagina 117). I test si concentrano sulla correttezza e conformità di:

- Entity Configuration (`.well-known/openid-federation`)

- Subordinate Statement restituiti da `/fetch`
- Endpoint del registro di federazione (`/list`, `/fetch`, `/trust_mark_status`, `/historical-jwks`)

Tutte le validazioni sono allineate con (OID-FED).

Test Case ID	Purpose	Description	Expected Result
FA_001	Discovery, Interoperability	Media type dell'Entity Configuration	`GET /.well-known/openid-federation` restituisce HTTP 200 con `Content-Type: application/entity-statement+jwt`.
FA_002	Security	Integrità della firma dell'Entity Configuration	L'Entity Configuration è un JWT firmato; la firma è verificabile usando una delle chiavi pubbliche della Federation Entity contenute in `jwks` secondo OID-FED.
FA_003	Interoperability	Parametri JOSE dell'header dell'Entity Configuration	L'header JOSE contiene `alg` (permesso), `kid` opzionale riferito a una chiave in `jwks`, e `typ` impostato a `entity-statement+jwt`.
FA_004	Security	Claim standard nell'Entity Configuration	Il payload include `iss` e `sub` entrambi uguali all'URL identificativo dell'Autorità di Federazione; include `iat` e `exp` come Unix timestamp validi; `exp` > `iat` e l'ora corrente < `exp`.
FA_005	Security, Interoperability	Parametri comuni dell'Entity Configuration	Il payload contiene `jwks` e `metadata` con l'oggetto `federation_entity` che include gli endpoint della Federazione pubblicati come da *Integrazione dell'Infrastruttura di Trust e del Registro* (pagina 120).

continues on next page

Tabella 20.25 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
FA_006	Security	Validità del materiale crittografico	Le voci in `jwks.keys[*]` usano algoritmi e dimensioni di chiave consentiti; le chiavi non sono scadute o revocate secondo `/historical-jwks`; ogni `kid` è univoco.
FA_007	Security	Tolleranza di validazione per `exp`	La validazione rifiuta l'Entity Configuration se `exp` è nel passato; PUÒ essere applicato uno skew massimo di 120 secondi quando si confrontano `iat`/`exp` con l'ora corrente.
FA_008	Security	Firma e durata del Subordinate Statement	`GET /fetch?sub={entity}` restituisce un JWT firmato (Subordinate Statement); header e payload sono validi secondo OID-FED; `iss` uguale all'Autorità emittente; `sub` uguale al soggetto richiesto; `iat`/`exp` validi.
FA_009	Interoperability	Schema del Subordinate Statement	Il Subordinate Statement contiene le chiavi pubbliche del subordinato (direttamente o per riferimento) e le eventuali policy di metadata o i metadata applicabili, conformi alla struttura della draft-43.
FA_010	Discovery	Elenco dei subordinati	`GET /list` restituisce un array JSON o un elenco JWT-wrapped degli identificativi dei subordinati correnti; il formato della risposta e il media type corrispondono a quanto pubblicato nei metadata; HTTP 200.

continues on next page

Tabella 20.25 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
FA_011	Security	Endpoint di stato dei Trust Mark	`GET /trust_mark_status?id={tm_id}&sub={entity}` restituisce lo stato corrente con HTTP 200 e un oggetto protetto in integrità (JWT se pubblicizzato). Trust Mark ID o soggetti sconosciuti restituiscono l'appropriato 4xx.
FA_012	Security	Endpoint delle chiavi storiche	`GET /historical-jwks` restituisce chiavi revocate/ scadute con le motivazioni della revoca. La struttura è valida come JWKS o JWKS incapsulato in JWT in base al media type pubblicizzato.
FA_013	Security	Propagazione della rotazione delle chiavi	Dopo la rotazione delle chiavi, `/.well-known/openid-federation` e `/historical-jwks` sono aggiornati in modo atomico o entro una finestra di propagazione massima documentata. La verifica con la nuova chiave ha esito positivo e la chiave precedente compare nell'elenco storico.
FA_014	Security	Disabilitare `alg": "none"` e algoritmi deboli	Qualsiasi Entity Configuration o Subordinate Statement con `alg": "none"` o un algoritmo non consentito è rifiutato; l'endpoint restituisce un errore appropriato (ad es., 400/422).
FA_015	Security	Risoluzione del `kid` e gestione mismatch	Se l'header JOSE include `kid`, questo corrisponde a una chiave in `jwks` corrente; in caso di mismatch, la verifica fallisce con errore chiaro.

continues on next page

Tabella 20.25 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
FA_016	Interoperability	Scoperta endpoint dai metadata	I metadata `federation_entity` dell'Entity Configuration includono URL funzionanti per `federation_list_endpoint`, `federation_fetch_endpoint`, `federation_trust_mark_status_endpoint` e `historical-jwks` (se pubblicato); ciascuno risponde secondo il proprio contratto.
FA_017	Security	Auto-coerenza issuer/subject	Per l'Entity Configuration, `iss == sub ==` URL dell'Autorità; per i Subordinate Statement, `iss` è l'URL dell'Autorità e `sub` è l'URL del subordinato; qualsiasi deviazione è rifiutata.
FA_018	Interoperability	Correttezza dei media type (fetch/list/status)	`/fetch`, `/list`, `/trust_mark_status` e `/historical-jwks` restituiscono i media type dichiarati nei metadata `federation_entity`; le risorse JWT-wrapped usano il `Content-Type` corretto (ad es., `application/entity-statement+jwt`, `application/jwk-set+jwt`).
FA_019	Security	Prevenzione replay dello Statement (`jti` opzionale)	Se `jti` è pubblicato, il riutilizzo entro la finestra di validità è rilevato e loggato o rifiutato secondo policy; altrimenti, l'unicità non è richiesta.
FA_020	Security	Applicazione delle policy di metadata (se usate)	Quando nel Subordinate Statement sono presenti policy di metadata, i metadata effettivi risultanti da policy + metadata sorgente sono conformi alle regole della draft-43; i conflitti sono risolti in modo deterministico.

Matrice di Test per Wallet Provider

Questa sezione fornisce l'insieme di casi di test per verificare la conformità di un'implementazione di Soluzione Wallet e Istanza del Wallet alle regole tecniche definite nell'ecosistema IT-Wallet. Il piano di test è basato sui requisiti estratti dalle seguenti Sezioni:

- *L'Infrastruttura di Trust* (pagina 117)
- *Soluzione Wallet* (pagina 287)
- *Emissione di Attestati Elettronici* (pagina 443)
- *Presentazione dell'Attestato Elettronico* (pagina 506)
- *Endpoints* (pagina 571)
- *Istanza dell'Applicazione Mobile* (pagina 705)

Nota

I casi di test in questo elenco variano di ambito: alcuni restano ad alto livello e rimandano al completamento dei flussi utente, alla conformità rispetto ai principi architetturali e di interoperabilità; altri ancora sono di basso livello, mirati a singoli requisiti dettagliati e verificano funzionalità specifiche.

Casi di Test per Backend del Fornitore del Wallet

Questa sezione elenca i casi di test dalle Sezioni:

- *Requisiti della Soluzione Wallet* (pagina 289)
- *Componenti della Soluzione Wallet* (pagina 294)
- *Istanza del Wallet* (pagina 298)
- *Entity Configuration del Fornitore di Wallet* (pagina 326)
- *Metadati della Soluzione Wallet* (pagina 329)
- e-Service PDND Wallet Provider Catalogue

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_001	Trust, Sicurezza	Pubblicazione Entity Configuration	Una richiesta HTTP GET all'endpoint `.well-known/openid-federation` recupera l'Entity Configuration del Fornitore del Wallet.
WP_002	Trust, Sicurezza	Firma Entity Configuration	L'Entity Configuration del Fornitore del Wallet è un JWT firmato che è conforme a OID-FED.
WP_002a	Trust, Interoperabilità	Parametro header `alg` del JWT Entity Configuration	L'header del JWT Entity Configuration contiene un `alg` impostato su un algoritmo di firma consentito (es. `ES256`) e non `none`.
WP_002b	Trust, Interoperabilità	Parametro header identificativo chiave del JWT Entity Configuration	L'header del JWT Entity Configuration contiene un `kid` uguale al thumbprint della chiave pubblica utilizzata per firmare l'Entity Configuration.
WP_002c	Trust, Interoperabilità	Parametro header tipo del JWT Entity Configuration	L'header del JWT Entity Configuration contiene `typ` impostato su `entity-statement+jwt`.
WP_002d	Trust, Interoperabilità	Claim payload issuer e subject dell'Entity Configuration	Il payload del JWT Entity Configuration contiene `iss` e `sub`, entrambi impostati sull'URL pubblico del Fornitore del Wallet.
WP_002e	Trust, Interoperabilità	Validità Entity Configuration	Il payload del JWT Entity Configuration contiene `iat` e `exp` come timestamp Unix validi.
WP_002f	Trust, Interoperabilità	Suggerimenti autorità Entity Configuration	Il payload del JWT Entity Configuration contiene `authority_hints` come array di URL validi di entità superiori immediate.
WP_002g	Trust, Interoperabilità	Chiavi di firma Entity Configuration	Il payload del JWT Entity Configuration contiene il parametro `jwks` avente uno o più JWK (**RFC 7517**) validi, ciascuno contenente chiavi pubbliche di firma del Fornitore del Wallet nel ruolo di Entità della Federazione .
WP_002h	Trust, Interoperabilità	Metadata Entity Configuration	Il payload del JWT Entity Configuration contiene un oggetto `metadata` che include il parametro `openid_wallet_provider` e opzionalmente il parametro `federation_entity`, ciascuno valorizzato seguendo il proprio schema.

continues on next page

Tabella 20.26 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_003	Trust, Interoperabilità	Utilizzo chiave metadata	Le chiavi pubbliche nell'oggetto JSON sono usate esclusivamente per la firma e/o cifratura quando l'Entità agisce come Fornitore del Wallet (es. per emettere la Wallet Instance Attestation (WIA)).
WP_004	Trust, Interoperabilità	Riferimento chiave metadata	Per riferire le chiavi pubbliche, l'oggetto JSON contiene esattamente uno dei seguenti claim: `jwks`, `jwks_uri`, o `signed_jwks_uri`.
WP_004a	Trust, Interoperabilità	JWKS per valore	Se `jwks` è presente, il suo valore contiene un elemento JWK (**RFC 7517**) valido.
WP_004b	Trust, Interoperabilità	JWKS per riferimento	Se `jwks_uri` è presente, il valore è un URL HTTPS a un JWKS che risulta risolvibile e valido.
WP_004c	Trust, Interoperabilità	URI JWKS firmato	Se `signed_jwks_uri` è presente, il valore è un URL HTTPS che punta a un JWT. Il payload di tale JWT è un JWK (**RFC 7517**) ed è firmato con una chiave dell'Entità di Federazione. La risorsa specificata da `signed_jwks_uri` è servita con il content type `application/jwk-set+jwt`. Un recupero riuscito di `signed_jwks_uri` restituisce lo stato `HTTP 200`.
WP_005	Trust, UX	Applicazione 2FA portale	Il portale del Fornitore di Wallet applica correttamente l'autenticazione a due fattori (2FA) prima di consentire agli Utenti di eseguire qualsiasi operazione.
WP_006	Revoca Wallet, Sicurezza	Vista stato Istanze del Wallet del portale	Il portale del Fornitore di Wallet consente correttamente agli Utenti di visualizzare lo stato corrente di tutte le Istanze del Wallet collegate al loro Account.
WP_007	Revoca Wallet, Sicurezza	Trigger revoca avviata dall'utente	Il Fornitore di Wallet implementa e supporta richieste di revoca avviate dagli Utenti tramite il suo portale.
WP_007a	Revoca Wallet, Sicurezza	Revoca istanza portale	Il portale del Fornitore di Wallet consente correttamente agli Utenti di revocare un'Istanza del Wallet specifica o tutte le Istanze del Wallet.

continues on next page

Tabella 20.26 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_008	Revoca Wallet, Sicurezza	Trigger revoca avviata dal Fornitore di Attestati Elettronici di Dati di Identificazione Personale	Il Fornitore di Wallet implementa e supporta richieste di revoca attivate dai Fornitori di PID o di IT-Wallet ID tramite il PDND Endpoint.
WP_009	Revoca Wallet, Sicurezza	Trigger revoca avviata dalle Autorità Legali	Il Fornitore di Wallet implementa e supporta richieste di revoca attivate dalle Autorità Legali o Organi di Supervisione (es. attività illegali).
WP_010	Trust, Interoperabilità	Meccanismo di revoca Istanza del Wallet	Quando il Fornitore di Wallet attiva la revoca per un'Istanza del Wallet specifica, in qualsiasi momento, quell'Istanza viene terminata e non può più eseguire alcuna funzione.
WP_011	Trust, Sicurezza	Verifica integrità Istanza del Wallet	Il Fornitore di Wallet valuta periodicamente l'integrità, l'autenticità e la genuinità dell'Istanza del Wallet utilizzando la Play Integrity API (Android) o App Attest (iOS).
WP_012	Ciclo di vita, Interoperabilità	Architettura componente backend	Il Backend del Fornitore di Wallet supporta tutti i componenti (Frontend, Interfaccia API, Gestione Ciclo di Vita Istanza del Wallet, e Trust & Sicurezza) come mostrato in *Figure of Wallet Solution High Level Architecture* (pagina 288).

Casi di Test per Istanza del Wallet

Questa sezione elenca i casi di test dalle Sezioni:

- Configuration of the Federation
- Trust Evaluation Mechanism
- *Requisiti della Soluzione Wallet* (pagina 289)
- *Componenti della Soluzione Wallet* (pagina 294)
- *Istanza del Wallet* (pagina 298)
- Error Handling for Wallet Instance Management
- Mobile Application Instance Initialization

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_013	Ciclo di vita, Interoperabilità	Architettura componente frontend	Istanza del Wallet supporta tutti i componenti (Interfaccia Utente, Gestione Ciclo di Vita, Fornitore di Attestato Elettronico, Presentazione, Backup/Ripristino, e Archiviazione Sicura) come mostrato in *Figure of Wallet Solution High Level Architecture* (pagina 288).
WP_014	Trust, Sicurezza	Profilo Keystore / WSCD	Il profilo implementativo attuale usa esclusivamente un Keystore hardware-backed interno locale (TEE o StrongBox su Android, Secure Enclave su iOS). L'Istanza del Wallet genera e memorizza le chiavi private dell'Utente in quel Keystore e non usa un keystore solo software. WSCD indica un dispositivo certificabile High (HSM o smart card, almeno Common Criteria EAL4+ AVA_VAN.5).
WP_014a	Trust, Sicurezza	Solo Keystore hardware-backed	I tentativi di generare o importare chiavi private dell'Utente in un keystore solo software sono rifiutati. Le operazioni crittografiche per WIA, KA, IT-Wallet ID e (Q)EAA usano il Local Internal Keystore.
WP_014b	Trust, Sicurezza	Non esportabilità delle chiavi private	Il Keystore o il WSCD non consente l'esportazione in chiaro delle chiavi private dell'Utente. Se il dispositivo dichiara la chiave privata come esportabile, il Fornitore del Wallet rifiuta la Key Attestation e non attiva l'Istanza del Wallet.
WP_014c	Trust, Sicurezza	Sole Control e autenticazione Utente	Il Keystore o il WSCD richiede l'autenticazione dell'Utente (sblocco wallet: PIN o biometrico) prima di qualsiasi firma o uso della chiave. L'Utente mantiene il controllo esclusivo delle chiavi private.
WP_014d	Trust, Sicurezza	Conservazione chiavi PID High vs IT-Wallet ID	Le chiavi di IT-Wallet ID e (Q)EAA POSSONO essere memorizzate nel Local Internal Keystore (eIDAS Substantial). Le chiavi di un PID a LoA High DEVONO essere memorizzate solo in un WSCD. Il profilo attuale non emette un PID High; un report TEE Protection Profile o equivalente non è accettato come prova di LoA High.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_014e	Trust, Sicurezza	Collocazione chiavi WIA	Le chiavi private associate alla WIA sono generate e memorizzate nello stesso Keystore o WSCD che la WIA attesta.
WP_014f	Trust, Sicurezza	Chiavi della Mobile Relying Party Instance	Una Mobile Relying Party Instance che non memorizza chiavi di identità dell'Utente PUÒ usare il Local Internal Keystore. Non è richiesto un WSCD certificabile High.
WP_015	Ciclo di vita, UX	Compatibilità Android/iOS	Istanza del Wallet si installa e opera con piena funzionalità su dispositivi Android e iOS ed è disponibile su Play Store e App Store.
WP_016	Trust, Issuance, Interoperabilità	Recupero e gestione metadata Trust Anchor	Istanza del Wallet recupera e aggiorna l'Entity Configuration del Trust Anchor prima dell'uso e la mantiene aggiornata.
WP_017	Trust, Issuance, Interoperabilità	Validare autenticità Trust Anchor	Istanza del Wallet confronta le chiavi pubbliche out-of-band con quelle nell'Entity Configuration del Trust Anchor e scarta le chiavi non corrispondenti, registra le discrepanze e utilizza solo le corrispondenze.
WP_018	Ciclo di vita, Trust, Sicurezza	Ristabilimento periodico del trust	Istanza del Wallet ottiene periodicamente e con successo una Wallet Instance Attestation (WIA) fresca dal suo Fornitore del Wallet.
WP_019	Trust, Sicurezza	Contenuto Wallet Instance Attestation (WIA)	La Wallet Instance Attestation (WIA) contiene tutti i claim e punti dati richiesti che attestano l'integrità e lo stato di sicurezza del dispositivo.
WP_019a	Trust, Sicurezza	Nessun attestato per Istanze del Wallet non verificate	Il Fornitore del Wallet rifiuta una richiesta di attestato da un'Istanza del Wallet che non supera i controlli di autenticità, integrità o genuinità.
WP_019b	Trust, Sicurezza	Associazione attestato-chiave effimera	La Wallet Instance Attestation (WIA) contiene una chiave pubblica effimera generata dall'Istanza del Wallet e Istanza del Wallet è in grado di dimostrare la proprietà della chiave privata corrispondente.
WP_020	Trust, Sicurezza	Firma Wallet Instance Attestation (WIA)	La Wallet Instance Attestation (WIA) è firmata digitalmente dal suo Fornitore del Wallet autorizzato, come ufficialmente elencato dall'Autorità di Registrazione supervisore.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_021	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Verifica sicurezza dispositivo	Istanza del Wallet non può essere attivata su un dispositivo che non soddisfa i requisiti minimi di sicurezza definiti dal Fornitore del Wallet.
WP_022	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Key Attestation dal Local Internal Keystore	Istanza del Wallet verifica e conferma la disponibilità delle Key Attestation API del Local Internal Keystore (StrongBox/TEE o Secure Enclave) e ottiene una Key Attestation (KA) per le chiavi generate in quel Keystore.
WP_022a	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	KA `key_storage` e `user_authentication`	Il JWT della KA imposta `key_storage` e `user_authentication` ai valori ISO 18045 coerenti con la resistenza al potenziale di attacco del Keystore o WSCD effettivo e del metodo di autenticazione Utente. Il Fornitore del Wallet documenta la mappatura usata per TEE, StrongBox e Secure Enclave. I valori NON DEVONO dichiarare `iso_18045_high` per un Local Internal Keystore.
WP_022b	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	KA per WSCD remoto o locale esterno	Se un profilo futuro usa un WSCD remoto (HSM) o un WSCD locale esterno (smart card, token), la KA è prodotta da quel WSCD, non dal TEE del telefono. È verificato il Sole Control dell'Utente sulle chiavi remote. Questo caso è fuori dal profilo attuale.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_023	Inizializzazione / Registrazione Wallet, Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Trust	Scoperta federazione Fornitore del Wallet	Istanza del Wallet utilizza con successo gli Endpoint di Federation (`.well-known/openid-federation`, `/fetch`) per recuperare metadata e configurazioni correnti del Fornitore del Wallet.
WP_024	Inizializzazione / Attivazione Wallet, Ciclo di vita, Sicurezza	Consenso utente	Durante l'inizializzazione del Wallet, il Fornitore del Wallet ottiene esplicitamente il consenso dell'Utente prima di creare un account Utente.
WP_025	Attivazione Wallet, Ciclo di vita, UX	Configurazione metodo sblocco Wallet	Istanza del Wallet richiede all'Utente di impostare il proprio metodo di sblocco preferito (PIN o biometrico), e quindi configura con successo il metodo scelto.
WP_026	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Coppia chiavi effimere per attestato	Per eseguire una Richiesta di Emissione della Wallet Instance Attestation (WIA), Istanza del Wallet genera con successo una nuova coppia di chiavi asimmetriche effimere.
WP_027	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica falla sicurezza dispositivo	Il Fornitore del Wallet verifica che il dispositivo soddisfi i suoi requisiti minimi di sicurezza e sia privo di falle di sicurezza note; se non è così, la Richiesta di Emissione della Wallet Instance Attestation (WIA) viene rifiutata.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_028	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Wallet Instance Attestation (WIA) a tempo limitato	La WIA rilasciata soddisfa *la durata della Wallet Instance Attestation* (pagina 290). Una formulazione limitata a «short-lived» non è sufficiente.
WP_029	Rilascio Wallet Instance Attestation (WIA), Modello Dati e Ciclo di vita, Interoperabilità	Busta risposta HTTP 200 / JSON	Dopo la validazione riuscita della Richiesta di Emissione della Wallet Instance Attestation (WIA), il Fornitore del Wallet restituisce un codice di stato 200 OK con Content-Type `application/json`, nel cui payload si trova l'array `wallet_attestations` contenente le Wallet Instance Attestation (WIA) definite in Wallet Instance Attestation (WIA) JWT.
WP_029a	Rilascio Wallet Instance Attestation (WIA), Modello Dati e Ciclo di vita, Sicurezza	Formato della Wallet Instance Attestation (WIA)	Il Fornitore del Wallet fornisce la Wallet Instance Attestation (WIA) formato JWT firmato dal Fornitore del Wallet, come definito in *Wallet Instance Attestation (WIA) JWT <wallet-provider-endpoint.html#wallet-instance-attestation-jwt>*.
WP_029b	Rilascio Wallet Instance Attestation (WIA), Modello Dati e Ciclo di vita, Sicurezza	Nessun PII nella Wallet Instance Attestation (WIA)	Il payload della Wallet Instance Attestation (WIA) non contiene informazioni personali sull'Utente.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_030	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica integrità Wallet Instance Attestation (WIA)	Istanza del Wallet verifica la firma della Wallet Instance Attestation (WIA) ricevuta.
WP_031	Rilascio Wallet Instance Attestation (WIA), Trust, Sicurezza	Verifica trust Wallet Instance Attestation (WIA)	Istanza del Wallet verifica e conferma che l'emittente della Wallet Instance Attestation (WIA) (cioè, Fornitore del Wallet) è un membro fidato della Federazione, altrimenti la rifiuta come non valida.
WP_032	Revoca Wallet, Ciclo di vita, Sicurezza	Revoca avviata dall'utente	L'Utente avvia con successo una revoca tramite la funzione di revoca integrata nell'Istanza del Wallet o tramite uno user agent esterno.
WP_032a	Revoca Wallet, Ciclo di vita, Interoperabilità	Login utente al portale con 2FA	Per eseguire una revoca dell'Istanza del Wallet, l'Utente accede con successo al portale web del Fornitore del Wallet con `2FA`.
WP_032b	Revoca Wallet, Ciclo di vita, UX	Selezione utente di un'Istanza del Wallet da revocare	L'Utente visualizza lo stato delle proprie Istanze del Wallet e ne sceglie una da revocare.
WP_033	Revoca Wallet, Ciclo di vita, Interoperabilità	Richiesta di Revoca Istanza del Wallet	Istanza del Wallet richiede la revoca di un'istanza specificata inviando una Richiesta di Revoca Istanza del Wallet all'Endpoint di Gestione dell'Istanza del Wallet.
WP_034	Revoca Wallet, Ciclo di vita, Sicurezza	Avvisi revoca tempestivi	Il Fornitore del Wallet informa l'Utente entro 24 ore di una revoca dell'Istanza del Wallet.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_034a	Revoca Wallet, Ciclo di vita, Sicurezza	Avvisi revoca out-of-band	Il Fornitore del Wallet notifica all'Utente una revoca dell'Istanza del Wallet attraverso un canale out-of-band (es. email o SMS).
WP_034b	Revoca Wallet, Ciclo di vita, Sicurezza	Notifica revoca chiara con guida riattivazione	La notifica di revoca spiega chiaramente il motivo della revoca e fornisce passaggi attuabili per la riattivazione, se applicabile.
WP_035	Ciclo di vita, Sicurezza	Gestione errori per gestione Istanza del Wallet	Quando si verifica un qualsiasi errore durante il flusso Registrazione/Inizializzazione dell'Istanza del Wallet, o durante richieste Recupero Stato e Revoca, il Fornitore del Wallet restituisce una risposta di errore all'Istanza del Wallet che è conforme agli standard definiti in **RFC 7231** (per il codice stato HTTP) e **RFC 7807** (per la descrizione del problema).
WP_035a	Ciclo di vita, Interoperabilità	Formato risposta errore standard	La risposta di errore restituita dal Fornitore del Wallet include un header `Content-Type` di `application/json` e un corpo JSON contenente i parametri obbligatori `error` e `error_description`.
WP_036	Ciclo di vita, Interoperabilità	Gestione richieste malformate (400)	Quando una richiesta è malformata o mancano parametri richiesti, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`400 Bad Request`) e con codice errore `bad_request`.
WP_037	Ciclo di vita, Interoperabilità	Errore validazione semantica (422)	Quando una richiesta è ben formata ma fallisce la validazione semantica, il Fornitore del Wallet restituisce una risposta opzionale con codice stato HTTP (`422 Unprocessable Content`) e con codice errore `validation_error`.
WP_038	Ciclo di vita, Interoperabilità	Errore server interno (500)	In caso di fallimento interno non gestito, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`500 Internal Server Error`) e con codice errore `server_error`.

continues on next page

Tabella 20.27 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_039	Ciclo di vita, Interoperabilità	Servizio non disponibile (503)	Quando il servizio è temporaneamente offline o incapace di gestire richieste, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`503 Service Unavailable`) e con codice errore `temporarily_unavailable`.
WP_040	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento inizializzazione (integrità dispositivo)	Quando una Richiesta di Inizializzazione Istanza del Wallet proviene da un dispositivo che non soddisfa i requisiti di sicurezza del provider, il provider restituisce una risposta `403 Forbidden` con codice errore `integrity_check_error`.
WP_041	Revoca Wallet, Ciclo di vita, Interoperabilità	Richiesta recupero non autorizzata (401)	Quando una richiesta recupero stato Istanza del Wallet è fatta senza credenziali di autenticazione valide, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`401 Unauthorized`) e con codice errore `unauthorized`.
WP_042	Revoca Wallet, Ciclo di vita, Interoperabilità	Richiesta recupero vietata (403)	Quando un Utente tenta di recuperare uno stato Istanza del Wallet per cui non ha permesso, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`403 Forbidden`) e con codice errore `forbidden`.
WP_043	Revoca Wallet, Ciclo di vita, Interoperabilità	Richiesta revoca non autorizzata (401)	Quando una Richiesta di Revoca di Istanza del Wallet è fatta senza credenziali di autenticazione valide, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`401 Unauthorized`) con codice errore `unauthorized`.
WP_044	Revoca Wallet, Ciclo di vita, Interoperabilità	Richiesta revoca vietata (403)	Quando un Utente tenta di revocare un'Istanza del Wallet per cui non ha permesso, il Fornitore del Wallet restituisce una risposta con codice stato HTTP (`403 Forbidden`) e con codice errore `invalid_request`.

Casi di Test per Fase di Issuance

Questa sezione elenca i casi di test dalle Sezioni:

- *Emissione di Attestati Elettronici* (pagina 443)
- Trust Evaluation Mechanism
- *Endpoint del Credential Issuer* (pagina 592)

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_045	Issuance, Interoperabilità	Scoperta Credential Issuer	Istanza del Wallet ottiene con successo la lista dei Credential Issuer fidati utilizzando il Catalogo degli Attestati Elettronici.
WP_045a	Trust, Issuance, Interoperabilità	Recupero Catalogo degli Attestati Elettronici	Istanza del Wallet invia con successo una richiesta HTTP GET al Catalogue Endpoint contenente il Catalogo degli Attestati Elettronici, utilizzando il media type `application/jose`.
WP_046	Issuance, Interoperabilità	Scoprire Credential Issuer dinamicamente dai metadata federazione	Istanza del Wallet utilizza con successo gli Endpoint di Federazione (`.well-known/openid-federation`, `/fetch`) per recuperare metadata e configurazioni correnti del Credential Issuer.
WP_046a	Trust, Issuance, Sicurezza	Validare Trust Chain completa del Credential Issuer al Trust Anchor	Istanza del Wallet costruisce e verifica la Trust Chain completa dal Credential Issuer attraverso Intermediari fino al Trust Anchor radice, assicurandosi che ogni firma sia valida.
WP_047	Issuance, Interoperabilità	Flusso Third-Party-Initiated: scansione e decodifica QR Credential Offer	In un flusso cross-device Third-Party-Initiated, l'Istanza del Wallet scansiona con successo il codice QR, e ottiene il parametro `credential_offer` o `credential_offer_uri`.
WP_048	Issuance, Interoperabilità	Flusso Third-Party-Initiated: parsing Credential Offer	In un flusso Third-Party-Initiated, l'Istanza del Wallet estrae e valida tutti i parametri richiesti presenti nell'oggetto Credential Offer.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_049	Issuance, Interoperabilità	Flusso Issuer-Initiated: controllo identificativo dell'Authorization Server	Istanza del Wallet legge i metadata `authorization_servers` del Credential Issuer e trova l'identificativo dell'Authorization Server configurato in `authorization_servers`.
WP_050	Issuance, Interoperabilità	Flusso Issuer-Initiated: validazione ID degli Attestati Elettronici	Istanza del Wallet verifica se ogni identificativo di Attestato Elettronico nei `credential_configuration_ids` è supportato dal Credential Issuer; qualsiasi identificativo non supportato non viene preso in considerazione.
WP_050a	Trust, Issuance, Interoperabilità	Applicazione Policy & Trust Marks	Istanza del Wallet verifica che il Credential Issuer sia autorizzato a rilasciare l'Attestato Elettronico richiesto attraverso l'applicazione delle policy di metadata e la verifica dei Trust Marks.
WP_050b	Trust, Issuance, Interoperabilità	Conferma ID supportato	Istanza del Wallet verifica che ogni `credential_configuration_id` richiesto appaia nei metadata `credential_configurations_supported` del Credential Issuer; ID mancanti causano il fallimento dell'offerta.
WP_051	Issuance, Interoperabilità	Richiesta Attestato Elettronico utilizzando Flusso OAuth 2.0 Code	Istanza del Wallet richiede con successo PID, IT-Wallet ID o (Q)EAA dal Fornitore corrispondente utilizzando il Flusso Authorization Code come in OpenID4VCI.
WP_052	Issuance, Interoperabilità	Richiesta PAR: costruzione payload	Istanza del Wallet genera un nuovo PKCE `code_verifier`, JWT PoP Wallet Instance Attestation (WIA), e valore `state`, quindi li avvolge in un Request Object firmato con la sua chiave privata (**RFC 9126**), e lo invia al PAR Endpoint.
WP_052a	Issuance, Interoperabilità	Generazione sicura `code_verifier` PKCE	Istanza del Wallet crea il `code_verifier` seguendo le raccomandazioni contenute in **RFC 7636**, come stringa casuale ad alta entropia utilizzando caratteri non riservati con lunghezza tra 43 e 128.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_052b	Issuance, Interoperabilità	Creazione JWT PoP Wallet Instance Attestation (WIA)	Istanza del Wallet genera il JWT PoP Wallet Instance Attestation (WIA) (per parametri `OAuth-Client-Attestation-PoP` come definito in OAUTH-ATTESTATION-CLIENT-AUTH) e include, nel payload, la chiave pubblica effimera referenziata nel `cnf.jwk` della Wallet Instance Attestation (WIA).
WP_052c	Issuance, Sicurezza	Firma JWT PoP Wallet Instance Attestation (WIA)	Istanza del Wallet firma il JWT PoP con la chiave privata effimera corrispondente alla chiave pubblica valorizzata nel parametro `cnf.jwk` della Wallet Instance Attestation (WIA).
WP_052d	Issuance, Interoperabilità	Specificare tipi degli Attestati Elettronici	Istanza del Wallet incorpora i tipi degli Attestati Elettronici nel Request Object utilizzando sia il parametro `scope` (per [OPENID4VC-HAIP]) sia il parametro `authorization_details` (per RAR **RFC 9396**).
WP_053	Issuance, Sicurezza	Authorization Request	Istanza del Wallet invia una PAR all'Authorization Endpoint del Credential Issuer utilizzando il `request_uri` e `client_id` ricevuti.
WP_053a	Issuance, Sicurezza	Uso di un unico `request_uri`	Istanza del Wallet invia l'Authorization Request con un `request_uri` appena ottenuto, e il Fornitore di Attestati Elettronici verifica che il `request_uri` non è stato usato precedentemente.
WP_054	Issuance, Interoperabilità	Validare l'Authorization Response: parametro richiesto	Istanza del Wallet verifica che l'Authorization Response contenga tutti i parametri richiesti (`code`, `state`, e `iss`); se vi sono parametri mancanti viene segnalato un errore.
WP_054a	Issuance, Sicurezza	Validare l'Authorization Response: verificare corrispondenza state	Istanza del Wallet confronta il valore `state` restituito con quello originariamente inviato nel Request Object (**RFC 6749**) e continua solo se corrispondono.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_054b	Issuance, Sicurezza	Validare l'Authorization Response: confermare identità Credential Issuer	Istanza del Wallet verifica che il parametro `iss` nell'Authorization Response corrisponda all'identificativo del Credential Issuer con cui ha iniziato la comunicazione (**RFC 9207**).
WP_055	Issuance, Interoperabilità	Inviare una Token Request	Istanza del Wallet invia una richiesta POST al Token Endpoint con URL-encoded body, includendo il `code` ricevuto (nell'Authorization Response), lo stesso `redirect_uri` (nel Request Object), e il `code_verifier`.
WP_055a	Issuance, Interoperabilità	Parametri prova Token Request	La Token Request porta le prove richieste negli header: un JWT prova DPoP, un JWT Wallet Instance Attestation (WIA), e JWT PoP dell'Istanza del Wallet (per `OAuth-Client-Attestation` e `OAuth-Client-Attestation-PoP` come definito in OAUTH-ATTESTATION-CLIENT-AUTH).
WP_055b	Issuance, Interoperabilità	Generare coppia chiavi/ prova DPoP	Istanza del Wallet genera una nuova coppia chiavi DPoP e un DPoP JWT seguendo le istruzioni fornite in **RFC 9449#section-4** per la Token Request al Credential Issuer.
WP_055c	Issuance, Sicurezza	Firmare prova DPoP	Istanza del Wallet firma il DPoP-JWT con la chiave privata DPoP generata precedentemente.
WP_055d	Issuance, Interoperabilità	Parametri PoP Wallet Instance Attestation (WIA) nella Token Request	`OAuth-Client-Attestation-PoP` è firmato utilizzando la chiave privata associata al Istanza del Wallet, dove la sua chiave pubblica correlata è fornita all'interno del Wallet Instance Attestation (WIA) (claim `cnf.jwk`).
WP_056	Issuance, Interoperabilità	Richiesta Attestato Elettronico	Istanza del Wallet invia una richiesta POST al Credential Endpoint includendo l'Access Token, il DPoP-JWT, tipo Attestato Elettronico (`credential_identifier`), e il PoP valido della chiave associata all'Attestato Elettronico.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_056a	Issuance, Sicurezza	Recuperare nonce per la dimostrazione di possesso chiave per Issuance di Attestato Elettronico	Istanza del Wallet invia una richiesta POST al Nonce Endpoint del Credential Issuer e ottiene un nuovo nonce, che viene poi utilizzato per generare una PoP del materiale crittografico che verrà associato all'Attestato Elettronico rilasciata.
WP_056b	Issuance, Sicurezza	Prova DPoP fresca	Istanza del Wallet genera una prova DPoP fresca con la stessa chiave utilizzata per il DPoP-JWT della Token Request, e in accordo con **RFC 9449#section-4**.
WP_056c	Issuance, Sicurezza	Abbinare la prova chiave Attestato Elettronico alla chiave DPoP	Istanza del Wallet include un oggetto JSON come valore del parametro `proofs` nella Credential Request. Al suo interno sono è presente l'array `jwt`. Quest'ultimo è valorizzato con uno o più JWT i cui parametro dell'header `jwk` rappresenta la prova il possesso della chiave pubblica referenziata nel DPoP-JWT della Token Request.
WP_057	Issuance, Interoperabilità	Richiesta di Attestati Elettronici multipli	Quando un Credential Issuer offre Attestati Elettronici multipli, Istanza del Wallet fa una Credential Request separata e per ogni Attestato Elettronico che intende ottenere.
WP_058	Issuance, Interoperabilità	Richiesta Attestato Elettronico in Batch	In caso di Batch Issuance, Istanza del Wallet invia una Credential Request in batch al Credential Endpoint che include: l'Access Token, un JWT prova DPoP, il tipo di Attestato Elettronico, e un parametro `proofs` contenente tutte le PoP delle chiavi private le cui pubbliche corrispondenti sono da associare all'Attestato Elettronico.
WP_058a	Issuance, Sicurezza	Batch Issuance: generazione chiavi	In caso di Batch Issuance, Istanza del Wallet genera un numero di nuove coppie chiavi uguale al valore `batch_size`.
WP_058b	Issuance, Interoperabilità	Batch Issuance: PoP delle chiavi Attestati Elettronici batch	In caso di Batch Issuance, Istanza del Wallet genera N prove di possesso (PoP) di chiave utilizzando il `c_nonce` fornito, una per ogni Attestato Elettronico nel batch, dove N uguale al valore `batch_size`.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_059	Issuance, Interoperabilità	Validare la Credential Response per parametri richiesti	Istanza del Wallet ispeziona il payload Credential Response, verificando che tutti i parametri PID, IT-Wallet ID o (Q)EAA obbligatori siano presenti e validi come definito in *Table of Credential Response* (pagina 616); se qualsiasi parametro è mancante o non valido, rifiuta la risposta con un errore.
WP_060	Issuance, Interoperabilità	Verificare tipo/ schema di un Attestato Elettronico	Istanza del Wallet recupera l'Attestato Elettronico rilasciata dal claim `credential` della risposta, verifica che il suo tipo corrisponda al tipo richiesto, e valida lo schema contro *Modello di Dati degli Attestati Elettronici* (pagina 379); se uno dei controlli fallisce, rifiuta l'Attestato Elettronico.
WP_061	Issuance, Sicurezza	Validare Trust Chain Credential Issuer	Istanza del Wallet verifica la Trust Chain del Fornitore di Attestati che ha emesso l'Attestato Elettronico usando i parametri presenti nel header dell'Attestato Elettronico stesso.
WP_062	Issuance, Interoperabilità	Verifica Attestato Elettronico per formato	Istanza del Wallet verifica se un Attestato Elettronico è in formato SD-JWT VC o mdoc-CBOR, e quindi esegue le verifiche appropriate. Se queste ultime falliscono, Istanza del Wallet rifiuta l'Attestato Elettronico.
WP_062a	Issuance, Sicurezza, Interoperabilità	Verificare integrità Attestato Elettronico SD-JWT	Istanza del Wallet ottiene i valori `alg` e `kid` dall'header SD-JWT, recupera la chiave pubblica corrispondente, e ne verifica la firma; se la verifica fallisce, l'Attestato Elettronico in formato SD-JWT viene rifiutato.
WP_062b	Issuance, Sicurezza, Interoperabilità	Verificare integrità Attestato Elettronico mdoc-CBOR	Istanza del Wallet estrae l'algoritmo di firma dall'header protetto del `COSE_Sign1` mdoc-CBOR, recupera la chiave pubblica del Credential Issuer dal `kid` o `x5chain` nell'header non protetto, e verifica la firma COSE sull'MSO; se non valida, l'Attestato Elettronico viene rifiutata.
WP_063	Issuance, Privacy	Consenso utente per archiviazione Attestato Elettronico	Istanza del Wallet richiede il consenso dell'Utente e archivia l'Attestato Elettronico solo dopo approvazione esplicita.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_064	Issuance, Interoperabilità	Gestione Notifiche	Istanza del Wallet invia una richiesta HTTP POST al Notification Endpoint con `Content-Type: application/json`.
WP_064a	Issuance, Interoperabilità	Parametri Notifica	Il payload Notification Request contiene il `notification_id` dalla Credential Response e il parametro `event` valorizzato `credential_accepted`, `credential_deleted`, o `credential_failure`; può anche includere un `event_description` ASCII valido.
WP_064b	Issuance, Privacy	Proteggere privacy Utente nel contenuto notifica	L'`event_description` contiene solo testo generico, neutrale per l'Utente e non rivela comportamento Utente o stato dispositivo.
WP_065	Issuance, Sicurezza	Gestire defererd Issuance	Istanza del Wallet valuta la Credential Response; se contiene sia `transaction_id` che `interval`, Istanza del Wallet riconosce il flusso come defererd Issuance.
WP_066	Issuance, Interoperabilità	Richiesta defererd Issuance dopo interval	Istanza del Wallet invia una Deferred Credential Response solo dopo che il `interval` richiesto è passato.
WP_066a	Issuance, Interoperabilità	Richiesta defererd Issuance con transaction_id	Istanza del Wallet invia una Deferred Credential Response come HTTP POST con `Content-Type: application/json`, e il corpo richiesta contiene il `transaction_id` richiesto.
WP_066b	Issuance, Interoperabilità	Richiesta defererd con Access Token ancora valido	Istanza del Wallet include l'Access Token esistente nella richiesta differita se il valore parametro `interval` è inferiore al tempo di scadenza impostato per l'Access Token.
WP_066c	Issuance, Interoperabilità	Nuovo Access Token DPoP-bound tramite Refresh	Quando l'Access Token esistente scadrebbe prima che la richiesta differita possa essere fatta, Istanza del Wallet ottiene un nuovo Access Token DPoP-bound tramite il flusso Refresh Token.
WP_067	Issuance, Interoperabilità	Nuovo flusso Issuance Attestato Elettronico	Quando Istanza del Wallet fallisce nel rinnovare un Access Token in scadenza, avvia un flusso Issuance Attestato Elettronico completamente nuovo.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_068	Issuance, Interoperabilità	Richiesta Refresh Token	Istanza del Wallet invia una richiesta POST al Token Endpoint del Credential Issuer con `grant_type=refresh_token`, un `refresh_token` valido, un header DPoP contenente un DPoP-JWT fresco, e un `OAuth-Client-Attestation` header contenente la Wallet Instance Attestation (WIA) un header `OAuth-Client-Attestation-JWT` contenente la PoP della chiave privata associata alla Wallet Instance Attestation (WIA).
WP_068a	Issuance, Sicurezza	Refresh Token: generare prove	Per una richiesta refresh Access Token, Istanza del Wallet genera un nuovo JWT DPoP e un nuovo PoP Wallet Instance Attestation (WIA), e li include nella richiesta.
WP_068b	Issuance, Interoperabilità	Refresh Token: mantenere associazione	Istanza del Wallet riutilizza la stessa chiave associata al PoP Wallet Instance Attestation (WIA) e alla DPoP-JWT dalla richiesta Access Token originale, assicurandosi che l'Access Token rinnovato rimanga crittograficamente associato al DPoP-JWT e valido.
WP_069	Issuance, Sicurezza	Controllo stato di un Attestato Elettronico	Istanza del Wallet verifica lo stato di ogni Attestato Elettronico archiviato recuperando e validando un Status List Token (per *Token di Status List* (pagina 437)).
WP_070	Issuance, Sicurezza	Flusso re-Issuance: rilevare necessità re-Issuance (aggiornare stato)	Istanza del Wallet aggiorna un Attestato Elettronico quando la Status List mostra `0x03` (`UPDATE`) o `0x0F` (`ATTRIBUTE_UPDATE`) per quell'Attestato Elettronico.
WP_071	Issuance, Sicurezza	Flusso re-Issuance: verificare validità Access Token per re-Issuance	Quando viene rilevato un aggiornamento per un Attestato Elettronico, Istanza del Wallet verifica la validità dell'Access Token associato. Se il token è valido, Istanza del Wallet procede con la re-Issuance dell'Attestato Elettronico.
WP_071a	Issuance, Sicurezza	Flusso re-Issuance: rinnovare Access Token scaduto	Se l'Access Token è scaduto ma è disponibile un Refresh Token valido, Istanza del Wallet avvia un flusso Refresh Token per ottenere un nuovo Access Token DPoP-bound, seguendo il Flusso Refresh Token.

continues on next page

Tabella 20.28 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_071b	Issuance, Sicurezza	Riavviare rilascio su scadenza completa	Istanza del Wallet ri-autentica l'Utente quando avvia un nuovo flusso di Issuance.
WP_072	Issuance, Sicurezza	Recuperare Attestato Elettronico aggiornato	Istanza del Wallet, con un Access Token DPoP-bound valido, invia una richiesta al Credential Endpoint e recupera con successo l'Attestato Elettronico aggiornato.
WP_073	Issuance, Sicurezza e Privacy	Eliminare vecchie Attestati Elettronici	Dopo aver archiviato un nuovo Attestato Elettronico, Istanza del Wallet elimina la versione precedente in modo che solo l'ultimo Attestato Elettronico rimanga archiviato.
WP_073a	Issuance, Sicurezza e Privacy	Eliminare vecchie Attestati Elettronici batch	Quando Istanza del Wallet riceve e archivia un nuovo batch dello stesso Attestato Elettronico con gli stessi claim, elimina gli Attestati Elettronici del batch precedente.
WP_074	Issuance, UX	Consenso su nuovo archivio Attestato Elettronico con aggiornamento `user_attribute`	Se l'aggiornamento Attestato Elettronico coinvolge cambiamenti ai valori degli Attributi dell'Utente (`attribute_update`), Istanza del Wallet richiede il consenso dell'Utente prima di archiviare il nuovo Attestato Elettronico aggiornato.
WP_075	Issuance, UX	Nessun consenso su nuovo archivio Attestato Elettronico senza aggiornamento `user_attribute`	Se l'aggiornamento Attestato Elettronico riguarda solo dei cambiamenti nei metadata dell'Attestato Elettronico (`update`), Istanza del Wallet archivia automaticamente il nuovo Attestato Elettronico aggiornato senza richiedere consenso aggiuntivo dall'Utente.

Casi di Test per Fase di Presentazione

Questa sezione elenca i casi di test dalla Sezione *Presentazione dell'Attestato Elettronico* (pagina 506), coprendo sia le fasi di presentazione **Flusso Remoto** che **Flusso di Prossimità**.

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_076	Flusso-remoto, Presentazione, Interoperabilità	Ottenere URL dell'Authorization Request nel flusso Same Device	Nel flusso Same Device, Istanza del Wallet ottiene con successo un URL ed estrae i seguenti parametri: `client_id`, `request_uri`, `state`, e `request_uri_method`.
WP_077	Flusso-remoto, Presentazione, Interoperabilità	Ottenere URL dell'Authorization Request nel flusso Cross Device	Nel flusso Cross Device, Istanza del Wallet scansiona e analizza con successo un Codice QR per estrarre i seguenti parametri: `client_id`, `request_uri`, `state`, e `request_uri_method`.
WP_078	Flusso-remoto, Trust, Presentazione, Interoperabilità	Scoperta identità Relying Party	Istanza del Wallet utilizza con successo gli Endpoint di Federazione (`.well-known/openid-federation`, `/fetch`) per recuperare metadata e configurazioni correnti della Relying Party.
WP_079	Flusso-remoto, Trust, Presentazione, Interoperabilità	Valutazione Trust Chain Relying Party	Istanza del Wallet valida con successo la Trust Chain della Relying Party (fornita staticamente o costruita attraverso un processo Federation Entity Discovery) dal Trust Anchor fino alla Relying Party stessa, confermando che la Relying Party è un membro riconosciuto e fidato della Federazione.
WP_080	Flusso-remoto, Trust, Presentazione, Interoperabilità	Controllo Trust Mark Relying Party	Istanza del Wallet valuta con successo qualsiasi Trust Mark incluso nell'Entity Configuration della Relying Party per assicurare che siano validi e indichino conformità alle policy della Federazione.
WP_081	Flusso-remoto, Trust, Presentazione, Sicurezza	Validare `request_uri`	Istanza del Wallet controlla che il `request_uri` sia nella lista di URI consentiti alla Relying Party (dai `openid_credential_verifier.request_uris` nella Trust Chain) e procede solo quando trova una corrispondenza esatta; se non viene trovata corrispondenza, rifiuta la richiesta.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_082	Flusso-remoto, Presentazione, Interoperabilità	GET Request Object	Se il `request_uri_method` nella richiesta iniziale è `get` o è assente, Istanza del Wallet invia una richiesta HTTP GET al `request_uri` Endopint della Relying Party, e recupera con successo un JWT Request Object firmato.
WP_083	Flusso-remoto, Presentazione, Interoperabilità	POST Request Object	Se il `request_uri_method` nella richiesta iniziale è `post`, Istanza del Wallet invia una richiesta HTTP POST al `request_uri` Endpoint della Relying Party, includendo i suoi metadata come parametri `wallet_metadata` e `wallet_nonce`, con content type `application/x-www-form-urlencoded`, e recupera con successo un JWT Request Object firmato.
WP_083a	Flusso-remoto, Presentazione, Interoperabilità	Costruire `wallet_metadata`	Istanza del Wallet formatta `wallet_metadata` come oggetto JSON secondo *Richiesta all'Endpoint URI Request* (pagina 522), includendo `vp_formats_supported`, `client_id_prefixes_supported`, `authorization_endpoint`, `response_types_supported` impostato su `["vp_token"]` quando presente.
WP_083b	Flusso-remoto, Presentazione, Privacy	Escludere PII in `wallet_metadata`	Il JSON `wallet_metadata` non contiene campi identificabili dall'utente o specifici del dispositivo (es. nomi, dettagli hardware).
WP_083c	Flusso-remoto, Presentazione, Sicurezza	Generare nonce replay	Un nuovo `wallet_nonce` viene generato e incluso nel payload della richiesta POST insieme al `wallet_metadata` per mitigare attacchi di replay.
WP_084	Flusso-remoto, Trust, Presentazione, Sicurezza	Recupero chiave pubblica Relying Party	Istanza del Wallet recupera la chiave pubblica corretta della Relying Party dal campo `metadata.openid_credential_verifier.jwks`, situato all'interno dell'Entity Configuration della Relying Party fornita dalla Trust Chain, utilizzando l'identificativo di chiave (`kid`) dall'header del Request Object.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_085	Flusso-remoto, Presentazione, Sicurezza	Verifica firma Request Object	Istanza del Wallet conferma l'integrità del Request Object firmato eseguendo con successo la validazione della firma crittografica usando la chiave pubblica della Relying Party.
WP_086	Flusso-remoto, Presentazione, Sicurezza	Abbinare `client_id` tra contesti	Istanza del Wallet conferma che il claim `iss` nel Request Object firmato abbia lo stesso `client_id` che ha originariamente estratto dall'URL dell'Authorization Request che il valore `sub` nell'Entity Configuration della Relying Party. Qualsiasi discrepanza causa il rifiuto dell Request Object.
WP_087	Flusso-remoto, Presentazione, Sicurezza	Controllare idoneità Relying Party	Istanza del Wallet autorizza la richiesta e procede con il flusso quando verifica che i metadata, le policy e un Trust Mark valido della Relying Party consentono a quest'ultima di richiedere l'Attestato Elettronico specificato.
WP_088	Flusso-remoto, Presentazione, Privacy	Consenso utente per divulgazione	Istanza del Wallet richiede il consenso dell'Utente presentando una schermata UI che mostra chiaramente l'identità verificata della Relying Party e un elenco di tutti gli Attributi richiesti.
WP_089	Flusso-remoto, Presentazione, UX	Gestire errori endpoint `request_uri`	Quando l'endpoint `request_uri` della Relying Party risponde con un errore, Istanza del Wallet mostra all'Utente un messaggio chiaro e leggibile, spiegando il motivo del fallimento e termina il flusso in modo sicuro.
WP_089a	Flusso-remoto, Presentazione, UX	Registrare errori `request_uri`	Istanza del Wallet registra internamente le informazioni dettagliate dell'errore.
WP_089b	Flusso-remoto, Presentazione, UX	Recuperare da errori `request_uri`	Per un errore recuperabile, come `server_error`, Istanza del Wallet richiede all'Utente un'azione di recupero specifica, come suggerire un retry o opzione "Scansiona QR di nuovo", se possibile.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_090	Flusso-remoto, Presentazione, Interoperabilità	Notificare Relying Party su richiesta non valida	Se il Request Object è non valido o ne fallisse la verifica, Istanza del Wallet invia una Authorization Error Response tramite HTTP POST al Response Endpoint (`response_uri`) della Relying Party.
WP_091	Flusso-remoto, Presentazione, Interoperabilità	Inviare l'Authorization Response	Se la verifica Request Object è valida, Istanza del Wallet invia una Authorization Response tramite HTTP POST al Response Endpoint (`response_uri`) della Relying Party.
WP_091a	Flusso-remoto, Presentazione, Sicurezza	Validare `response_uri`	Prima di inviare l'Authorization Response, Istanza del Wallet conferma che il `response_uri` corrisponda esattamente a uno dei `response_uris` attestati della Relying Party nei suoi metadata; una mancata corrispondenza interrompe il flusso.
WP_092	Flusso-remoto, Presentazione, Sicurezza	Cifrare l'Authorization Response	Istanza del Wallet cifra il JWT dell'Authorization Response per Sezione 8.3 di [OpenID4VP] utilizzando la chiave pubblica della Relying Party.
WP_093	Flusso-remoto, Presentazione, Sicurezza	Costruire `vp_token` con `state`	Il payload JWT dell'Authorization Response include il valore `state` originale e un parametro `vp_token` contenetne un oggetto le cui coppie chiave-valore corrispondono agli Attestati Elettronici richiesti.
WP_093a	Flusso-remoto, Presentazione, Interoperabilità	Includere presentazioni firmate	All'interno di quel `vp_token`, ci sono gli Attestati Elettronici richiesti in formato SD JWT VC.
WP_093b	Flusso-remoto, Presentazione, Sicurezza	Aggiungere Key Binding JWT	Istanza del Wallet genera per ogni singola presentazione SD-JWT all'interno del `vp_token`, il corrispondente Key Binding JWT.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_093c	Flusso-remoto, Presentazione, Sicurezza	Formato Key Binding JWT	L'header JOSE di ogni Key Binding JWT generato includeil parametro `typ` valorizzato con `kb+jwt` e un parametro `alg` valorizzato con uno degli algoritmi di firma supportati. Il payload include i parametri `iat`, `aud`, `nonce`, e `sd_hash` come definito in OpenID4VP.
WP_094	Flusso-remoto, Presentazione, Interoperabilità	Eseguire redirect user-agent	Dopo aver ricevuto il `redirect_uri`, Istanza del Wallet esegue con successo un redirect dello user-agent al `redirect_uri` fornito dalla Relying Party, così la Relying Party può riprendere l'interazione sullo stesso dispositivo che ha iniziato il flusso.
WP_094a	Flusso-remoto, Trust, Presentazione, Sicurezza	Validare `redirect_uri`	Istanza del Wallet verifica che il `redirect_uri` corrisponda esattamente a uno degli URI elencati nei metadata `redirect_uris` della Relying Party. Se non corrisponde, interrompe il flusso.
WP_095	Flusso-prossimità, Presentazione, Sicurezza	Supportare recupero supervisionato/ non supervisionato	Istanza del Wallet supporta una presentazione di Attestato Elettronico utilizzando il metodo Supervised Device Retrieval per uno scenario di verifica in presenza con supervisione umana e tramite Device Retrieval (non supervisionato) per verifica automatizzata senza supervisione umana.
WP_096	Flusso-prossimità, Presentazione, Sicurezza	Supporto meccanismi Device Retrieval	Istanza del Wallet supporta i meccanismi Device Retrieval tramite Bluetooth Low Energy (BLE) o NFC.
WP_096a	Flusso-prossimità, Presentazione, Sicurezza	Applicare solo device retrieval	Istanza del Wallet rifiuta qualsiasi richiesta di iniziare un flusso prossimità utilizzando il meccanismo Server Retrieval.
WP_096b	Flusso-prossimità, Presentazione, Sicurezza	Supportare device retrieval BLE/NFC (condizionale)	Istanza del Wallet completa con successo il flusso di recupero BLE o NFC. Il supporto BLE è obbligatorio, mentre NFC è RACCOMANDATO. Almeno un metodo di device retrieval (BLE o NFC) deve essere supportato; il fallimento costituisce non conformità.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_097	Flusso-prossimità, Presentazione, UX	Meccanismi Device Engagement supportati	Istanza del Wallet supporta Device Engagement basato su codice QR o NFC Connection Handover.
WP_097a	Flusso-prossimità, Presentazione, Sicurezza	Supporto QR/NFC engagement (condizionale)	Istanza del Wallet completa una transazione di recupero dati sia via BLE che via NFC (quando l'hardware è disponibile). Il supporto QR è obbligatorio, mentre NFC è RACCOMANDATO. Almeno un metodo di engagement (QR o NFC) deve essere supportato; il fallimento costituisce non conformità.
WP_098	Flusso-prossimità, Presentazione, Sicurezza	Autenticazione Relying Party	Istanza del Wallet supporta ed esegue l'Autenticazione dell'Istanza Relying Party in conformità al processo di Reader Authentication ISO18013-5.
WP_099	Flusso-prossimità, Presentazione, Interoperabilità	Supporto mDL domestico	Istanza del Wallet supporta e processa senza errore Attestati Elettronici mDL utilizzando le definizioni standard ISO18013-5 ed eventuali aggiunte nel namespace domestico di IT Wallet (vedi mdoc-CBOR Credential Format per maggiori dettagli).
WP_100	Flusso-prossimità, Presentazione, Sicurezza	Autenticazione WSCA	Istanza del Wallet richiede all'Utente di eseguire un'autenticazione basata su WSCA, direttamente o sbloccando l'applicazione, e non procede con il flusso prossimità finché non ha successo.
WP_101	Flusso-prossimità, Presentazione, Sicurezza	Generare coppia chiavi EC effimere	Istanza del Wallet genera con successo una nuova coppia di chiavi curva ellittica effimere (per la cipher suite ISO18013-5 scelta).
WP_102	Flusso-prossimità, Presentazione, Interoperabilità	Codifica CBOR dei dati Device Engagement	DeviceEngagement è codificato CBOR e contiene i parametri `Version`, `Security`, `Capabilities` e `OriginInfos` (può essere vuoto). Per QR engagement include anche `DeviceRetrievalMode-BLEOptions` e/o `DeviceRetrievalMode-NFCOptions`. Per NFC engagement, tali opzioni sono omesse.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_102a	Flusso-prossimità, Presentazione, Interoperabilità	Device Engagement via QR	L'URI mdoc: QR codifica un Device Engagement valido in CBOR (per Sezione 9.1 di [ISO18013-5]) usando base64url-without-padding (**RFC 4648**).
WP_102b	Flusso-prossimità, Presentazione, Interoperabilità	Verificare parametro Security	Il parametro `Security` all'interno dei dati DeviceEngagement contiene un identificativo cipher suite supportato e la chiave pubblica effimera dell'Istanza del Wallet per ISO18013-5 Tabella 22.
WP_102c	Flusso-prossimità, Presentazione, Interoperabilità	Verificare parametro `DeviceRetrievalMode-BLEOptions`	Il parametro `DeviceRetrievalMode-BLEOptions` all'interno dei dati DeviceEngagement indica Central Client Mode e porta l'UUID dispositivo corretto.
WP_102d	Flusso-prossimità, Presentazione, Interoperabilità	Verificare parametro `DeviceRetrievalMode-NFCOptions`	Il parametro DeviceRetrievalMode-NFCOptions dichiara ruolo supportato (PICC per Istanza del Wallet) e dimensioni massime comando/risposta APDU per mappatura ISO.
WP_102e	Flusso-prossimità, Presentazione, Interoperabilità	Verificare parametro `Capabilities`	Il parametro `Capabilities` all'interno dei dati DeviceEngagement imposta correttamente i parametri `HandoverSessionEstablishmentSupport` e `ReaderAuthAllSupport`.
WP_102f	Flusso-prossimità, Presentazione, Interoperabilità	Verificare parametro `OriginInfos`	Se presente, il parametro `OriginInfos` nei dati DeviceEngagement è correttamente codificato come array (che può essere vuoto).
WP_103	Flusso-prossimità, Presentazione, UX	Device Engagement via NFC (Connection Handover)	Istanza del Wallet espone DeviceEngagement tramite NFC Connection Handover (Statico o Negoziato).

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_103a	Flusso-prossimità, Presentazione, Interoperabilità	NFC Connection Handover (Statico)	Istanza del Wallet agisce come NFC Tag (Tipo 4) e fornisce un Handover Select con almeno un Alternative Carrier Record e relativo Carrier Configuration Record; un Auxiliary Data Record trasporta il DeviceEngagement (tipo `iso.org:18013:deviceengagement`, id "mdoc").
WP_103b	Flusso-prossimità, Presentazione, Interoperabilità	NFC Connection Handover (Negoziato)	Istanza del Wallet espone il servizio `urn:nfc:sn:handover`; su Handover Request restituisce un Handover Select con esattamente un carrier selezionato e il DeviceEngagement ausiliario.
WP_103c	Flusso-prossimità, Presentazione, Sicurezza, Interoperabilità	`Session Establishment` anticipato via TNEP	Se `HandoverSessionEstablishmentSupport` è `true` nel DeviceEngagement, Istanza del Wallet accetta il `SessionEstablishment` anticipato sul servizio TNEP annunciato e successivamente verifica che corrisponda a quello ricevuto durante il recupero dati.
WP_103d	Flusso-prossimità, Presentazione, Interoperabilità	Codifica Carrier Configuration Record	Istanza del Wallet include correttamente un Carrier Configuration Record per ogni carrier supportato: per NFC (tipo `iso.org:18013:nfc`, ID `nfc`) per Sezione 9.2.2 di [ISO18013-5]; per BLE per Sezione 11.1.2 di [ISO18013-5].
WP_103e	Flusso-prossimità, Presentazione, Interoperabilità	Auxiliary Data Record per Device Engagement	Istanza del Wallet fornisce un Auxiliary Data Record che trasporta il DeviceEngagement (tipo `iso.org:18013:deviceengagement`, ID `mdoc`) utilizzando NFC Forum external type (0x04); ogni Alternative Carrier Record DEVE fare riferimento a questo record.
WP_103f	Flusso-prossimità, Presentazione, Interoperabilità	Alternative Carrier Records	In Static Handover, l'Handover Select include uno o più Alternative Carrier Records; in Negotiated Handover contiene esattamente un carrier selezionato. L'Alternative Carrier NFC referenzia il Carrier Configuration Record `nfc`.
WP_103g	Flusso-prossimità, Presentazione, Sicurezza	Mismatch di `Session Establishment` anticipato	Se il `SessionEstablishment` anticipato è ricevuto nel Negotiated Handover, Istanza del Wallet verifica la corrispondenza con quello ricevuto nel recupero dati; in caso di mancata corrispondenza termina il flusso.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_103h	Flusso-prossimità, Presentazione, Interoperabilità	Assenza di Session Establishment anticipato	Se il `SessionEstablishment` anticipato non è ricevuto durante il Negotiated Handover, Istanza del Wallet procede con il Device Retrieval normalmente (nessun errore).
WP_104	Flusso-prossimità, Presentazione, Sicurezza	Derivare chiave di sessione	Istanza del Wallet deriva con successo le chiave di sessione eseguendo il key-agreement negoziato con la propria chiave privata effimera e la chiave pubblica effimera della Relying Party.
WP_105	Flusso-prossimità, Presentazione, Sicurezza	Decifrare & verificare `Session Establishment`	Istanza del Wallet decifra il messaggio `Session Establishment` usando la chiave di sessione derivata e valida la firma della Relying Party nel parametro `readerAuthAll`.
WP_106	Flusso-prossimità, Presentazione, Sicurezza	Validare contenuti `Session Establishment`	Istanza del Wallet verifica che `Session Establishment` includa la chiave pubblica della Relying Party e la richiesta di specifici attributi.
WP_107	Flusso-prossimità, Presentazione, Privacy	Richiedere consenso agli attributi	Istanza del Wallet decifra e mostra gli attributi richiesti all'Utente e procede solo dopo approvazione esplicita di quest'ultimo.
WP_107a	Flusso-prossimità, Presentazione, Privacy	Mostrare certificato Relying Party	Istanza del Wallet mostra all'Utente il Certificato di Registrazione della Relying Party prima di ottenere il consenso dell'Utente.
WP_108	Flusso-prossimità, Presentazione, Interoperabilità	Recuperare Attestati Elettronici mdoc	Istanza del Wallet recupera gli Attestati Elettronici richiesti e li prepara per l'mdoc response.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_109	Flusso-prossimità, Presentazione, Interoperabilità	Preparare la risposta mdoc	Istanza del Wallet costruisce il messaggio `SessionData` (risposta mdoc) in CBOR includendo un array `documents` con gli Attestati Elettronici richiesti.
WP_110	Flusso-prossimità, Presentazione, Sicurezza, Interoperabilità	Autenticazione mdoc	Istanza del Wallet firma correttamente i dati di autenticazione `deviceSigned` per ogni Attestato Elettronico presentato come spiegato nella Sezione 12.4 di [ISO18013-5].
WP_111	Flusso-prossimità, Presentazione, Interoperabilità	Validare struttura risposta mdoc	La risposta mdoc è un oggetto CBOR con parametro `version` alla radice e ogni documento in `documents` contenente `docType`.
WP_111a	Flusso-prossimità, Presentazione, Sicurezza	Validare parametro `deviceSigned`	Ogni documento include un `nameSpaces` (anche vuoto) e un `deviceAuth`. Quest'ultimo è un JSON struttura avente un parametro `deviceSignature` contenente un Array `COSE_Sign1` che autentica il dispositivo.
WP_112	Flusso-prossimità, Presentazione, Sicurezza	Cifrare `SessionData`	Istanza del Wallet cifra il `SessionData` con la chiave di sessione derivata.
WP_112a	Flusso-prossimità, Presentazione, Interoperabilità	Trasmettere messaggi su BLE	Su BLE (usando le caratteristiche GATT ISO), Istanza del Wallet riceve correttamente il `SessionEstablishment` cifrato dalla Relying Party e trasmette il proprio `SessionData` cifrato (e gli eventuali codici di stato/terminazione).
WP_112b	Flusso-prossimità, Presentazione, Interoperabilità	Trasmettere messaggi su NFC	Su NFC (usando il flusso APDU), Istanza del Wallet riceve correttamente il `SessionEstablishment` cifrato dalla Relying Party e trasmette il proprio `SessionData` cifrato, gestendo senza errori frammentazione e status word.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_112c	Flusso-prossimità, Presentazione, Interoperabilità	Coerenza UUID/MTU BLE	Gli UUID di servizio/caratteristiche BLE e l'MTU effettivo sono coerenti con quanto descritto in DeviceEngagement (e/o Carrier Configuration) e i messaggi sono segmentati di conseguenza.
WP_112d	Flusso-prossimità, Presentazione, Sicurezza	Caratteristica BLE Ident (opzionale)	Se implementata, Istanza del Wallet verifica il valore Ident della Relying Party prima del recupero dati e disconnette BLE in caso di mancata corrispondenza, per Sezione 11.1.3 di [ISO18013-5].
WP_112e	Flusso-prossimità, Presentazione, Interoperabilità	Gestione SELECT AID NFC	Istanza del Wallet espone l'*AID* e, su SELECT *APDU*, restituisce *FCI* valido con *SW1/SW2* corretti; seguono scambi ENVELOPE/GET RESPONSE.
WP_112f	Flusso-prossimità, Presentazione, Interoperabilità	Coerenza dimensioni APDU NFC	Le dimensioni comando/risposta *APDU* (inclusa frammentazione e gestione *SW1=61*) sono coerenti con le dimensioni massime descritte (Carrier Configuration / NFCOptions) durante il Device Engagement.
WP_113	Flusso-prossimità, Presentazione, Sicurezza	Terminare sessione / timeout inattività	Istanza del Wallet termina automaticamente la sessione se non vengono inviati/ricevuti messaggi di sessione per l'intervallo di timeout configurato.
WP_113a	Flusso-prossimità, Presentazione, Sicurezza	Terminare sessione / nessuna ulteriore richiesta	Istanza del Wallet termina la sessione quando essa o la Relying Party hanno concluso lo scambio dati.
WP_113b	Flusso-prossimità, Presentazione, Sicurezza	Inviare segnale di terminazione (BLE)	Istanza del Wallet invia il comando End (o codice di terminazione) su BLE per segnalare la chiusura della sessione.

continues on next page

Tabella 20.29 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
WP_113c	Flusso-prossimità, Presentazione, Sicurezza	Inviare terminazione su NFC	Istanza del Wallet segnala la fine tramite uno status code in `SessionData` e/o completa lo scambio APDU secondo ISO; non sono accettate ulteriori APDU per la sessione.
WP_114	Flusso-prossimità, Presentazione, Sicurezza	Distruggere chiave di sessione	Alla chiusura della sessione, Istanza del Wallet cancella chiave di sessione e materiale effimero correlato da memoria e archiviazione.
WP_114a	Flusso-prossimità, Presentazione, Sicurezza	Chiudere canale di comunicazione	Istanza del Wallet chiude il canale attivo disconnettendo il link BLE o terminando lo scambio APDU NFC; non rimangono canali aperti.

Casi di Test per Eliminazione Attributi dell'Utente Lato Relying Party

Questa sezione elenca i casi di test dalle Sezioni:

- *Eliminazione degli Attributi dell'Utente* (pagina 316)
- Relying Party Provider Backend Erasure Endpoint

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_115	Eliminazione Attributi, Ciclo di vita, Privacy	Funzione eliminazione attributi	L'UI dell'Istanza del Wallet fornisce funzioni per consentire all'Utente di richiedere l'eliminazione di attributi, visualizzare log transazioni, e vedere un elenco delle Relying Party verso le quali ha presentato degli Attributi.
WP_115a	Eliminazione Attributi, Ciclo di vita, Privacy	Log transazioni per eliminazione attributi	La vista dei log delle transazioni mostra le Relying Party in possesso degli attributi dell'Utente.

continues on next page

Tabella 20.30 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_116	Eliminazione Attributi, Ciclo di vita, Sicurezza	Validazione metadata Relying Party per eliminazione	Quando un Utente seleziona una Relying Party per l'eliminazione degli Attributi, Istanza del Wallet recupera e valida i suoi metadata di Federazione, confermando la presenza di un Endpoint di Cancellazione della Relying Party.
WP_117	Eliminazione Attributi, Ciclo di vita, Interoperabilità	Richiesta Cancellazione	Istanza del Wallet costruisce correttamente, e invia una Richiesta Cancellazione valida destinata all'Endpoint di Cancellazione della Relying Party.
WP_117a	Eliminazione Attributi, Ciclo di vita, Privacy	Registrazione Richiesta Cancellazione	Una voce nel log viene creata per ogni Richiesta Cancellazione, contenente il timestamp, l'identificativo della Relying Party, e gli Attributi specifici la cui eliminazione è richiesta.
WP_118	Eliminazione Attributi, Ciclo di vita, UX	Reindirizzamento utente e callback per cancellazione	Istanza del Wallet reindirizza con successo l'Utente all'Endpoint di Cancellazione della Relying Party e riceve la Risposta Cancellazione tramite un opportuno meccanismo di callback.
WP_119	Eliminazione Attributi, Ciclo di vita, Interoperabilità	Gestisce risposta eliminazione/ errore	Istanza del Wallet processa correttamente sia le risposte di successo che di errore per la Risposta Cancellazione dalla Relying Party.
WP_119a	Eliminazione Attributi, Ciclo di vita, UX	Notifica utente su cancellazione	Istanza del Wallet mostra una notifica chiara all'Utente indicando il successo o fallimento della Richiesta di Cancellazione.

Casi di Test per Backup e Ripristino degli Attestati Elettronici

Questa sezione elenca i casi di test dalla Sezione *Backup e Ripristino* (pagina 320).

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_120	Backup e Ripristino, UX	Avvio backup Attestato Elettronico	L'Utente attiva con successo l'operazione backup Attestato Elettronico sull'Istanza del Wallet.
WP_120a	Backup e Ripristino, Sicurezza	Visualizzazione frase-chiave casuale	Istanza del Wallet invoca le API backup, seleziona 10 frasi-chiave casuali da un elenco pre-generato e le mostra all'Utente.
WP_120b	Backup e Ripristino, Sicurezza	Istruzione archiviazione sicura frase-chiave	Istanza del Wallet mostra un'istruzione chiara per l'Utente per archiviare in modo sicuro le frasi-chiave generate.
WP_121	Backup e Ripristino, Sicurezza	Derivazione chiave da frasi-chiave	Istanza del Wallet deriva una chiave crittografica dalle frasi-chiave dell'Utente utilizzando una funzione derivazione chiave (es. `PBKDF2`, `Bcrypt`, `Scrypt`).
WP_121a	Backup e Ripristino, Sicurezza	Configurazione funzione derivazione chiave	Se `PBKDF2` con `SHA-256` è utilizzato, la funzione è configurata con un conteggio iterazioni di almeno `600,000`.
WP_122	Backup e Ripristino, Interoperabilità	Creazione JWT backup firmato	Istanza del Wallet crea un file backup contenente un JWT firmato che incapsula i dati backup.
WP_122a	Backup e Ripristino, Interoperabilità	Header JOSE per JWT backup	L'header JWT backup contiene l'`alg` utilizzato per firmare il backup e un `typ` di `wallet-unit-credentials-backup+jwt`.
WP_122b	Backup e Ripristino, Interoperabilità	Contenuti payload JWT backup	Il payload JWT backup include i claim richiesti: `timestamp`, `wallet_provider_id`, `wallet_instance_version`, `wallet_attestation`, e `credentials_backup`.
WP_122c	Backup e Ripristino, Sicurezza	Cronologia transazioni opzionale nel JWT backup	Il claim `credentials_backup` contiene la cronologia transazioni per ogni Attestato Elettronico, se supportato.

continues on next page

Tabella 20.31 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_122d	Backup e Ripristino, Sicurezza	Identificatori Attestato Elettronico nel JWT backup	Il claim `credentials_backup` contiene voci chiave dall'identificativo Fornitore di Attestato Elettronico, e ogni voce elenca uno o più `credential_configuration_ids` corrispondenti agli Attestati Elettronici rilasciati.
WP_123	Backup e Ripristino, Sicurezza	Firma JWT backup con chiave attestata	Istanza del Wallet firma il JWT backup con la chiave privata corrispondente alla chiave pubblica trovata nel claim `cnf` del JWT Wallet Instance Attestation (WIA).
WP_123a	Backup e Ripristino, Sicurezza	Controllo validità attestato per JWT backup	Prima di firmare il JWT backup, Istanza del Wallet verifica la validità del proprio Wallet Instance Attestation (WIA).
WP_124	Backup e Ripristino, Sicurezza	Cifratura file backup con chiave Utente	Istanza del Wallet cifra il backup utilizzando la chiave derivata dalle frasi-chiave dell'Utente.
WP_125	Backup e Ripristino, UX	Prompt selezione posizione archiviazione sicura	Istanza del Wallet richiede all'Utente di selezionare una posizione per salvare il file backup cifrato, offrendo opzioni come archiviazione locale o cloud.
WP_126	Backup e Ripristino, UX	Avvio ripristino Attestato Elettronico	L'Utente attiva con successo l'operazione ripristino degli Attestati Elettronici sull'Istanza del Wallet.
WP_127	Backup e Ripristino, UX	Selezione file backup/frase-chiave per ripristino	L'Utente seleziona con successo un file backup dall'archiviazione e inserisce le loro frasi-chiave di recupero.
WP_128	Backup e Ripristino, Interoperabilità	Estrazione Wallet Instance Attestation (WIA) da JWT backup	Istanza del Wallet estrae la Wallet Instance Attestation (WIA) dal claim `wallet_attestation` del JWT backup trovato all'interno del file backup.
WP_128a	Backup e Ripristino, Sicurezza	Ignorare scadenza Wallet Instance Attestation (WIA) durante ripristino	Il processo ripristino continua con successo anche se la Wallet Instance Attestation (WIA) nel file backup è scaduto.

continues on next page

Tabella 20.31 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_129	Backup e Ripristino, Sicurezza	Verifica firma JWT backup	Istanza del Wallet verifica con successo la firma del JWT backup utilizzando la chiave pubblica contenuta nel parametro `cnf.jwk` del JWT valorizzato nel parametro `wallet_attestation`.
WP_130	Backup e Ripristino, Interoperabilità	Emissione di Attestato Elettronico da backup ripristinato	Per ogni voce Attestato Elettronico nel JWT backup, Istanza del Wallet estrae con successo l'URL Fornitore di Attestati, e il `credential_configuration_id`.
WP_130a	Backup e Ripristino, Interoperabilità	Recuperare metadata Fornitore di Attestato Elettronico	Per ogni URL Fornitore di Attestato Elettronico, Istanza del Wallet recupera con successo i metadata del Fornitore di Attestato Elettronico corrispondenti.
WP_130b	Backup e Ripristino, Interoperabilità	Richiesta emissione Attestato Elettronico	Per ogni Attestato Elettronico, Istanza del Wallet avvia con successo un nuovo Wallet-Initiated Authorization Code Issuance Flow legato alla nuova chiave dell'Istanza del Wallet (nuova Associazione Crittografica con l'Utente), e non un rinnovo del token né una richiesta di Re-issuance Flow.

Casi di Test Opzionali per Istanza del Wallet

Questa sezione elenca i casi di test dalle Sezioni:

- *Istanza del Wallet* (pagina 298)
- Request-Specific Error Responses
- Mobile Application Instance Initialization

Questi casi di test sono opzionali e sono stati progettati per l'implementazione IT Wallet, gestita da PagoPA. Sono destinati agli implementatori che devono verificare la compatibilità con le caratteristiche specifiche di questo modello di riferimento.

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_131	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Richiesta nonce per protezione da replay attack	Istanza del Wallet richiede e riceve un nonce monouso, di breve durata dal Nonce Endpoint del Fornitore del Wallet.
WP_132	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Generazione/ archiviazione coppia chiavi EC hardware-backed	Istanza del Wallet genera con successo una coppia chiavi crittografiche hardware-backed fresca e archivia il suo `hardware_key_tag` associato nell'archiviazione sicura locale.
WP_132a	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Eliminazione chiave pre-esistente durante inizializzazione	Istanza del Wallet controlla la presenza di Cryptographic Hardware Keys pre-esistenti durante l'inizializzazione, se presenti, le elimina con successo.
WP_133	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Invio Richiesta di Inizializzazione Wallet	Istanza del Wallet invia con successo al Fornitore del Wallet una richiesta di Inizializzazione con i parametri richiesti `nonce`, `key_attestation`, e `hardware_key_tag`.
WP_133a	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Calcolo hash SHA-256	Istanza del Wallet calcola con successo un digest SHA-256 (`client_data_hash`) sul `nonce`, il `hardware_key_pub`, e il `hardware_key_tag`,

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_133b	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Key Attestation	Istanza del Wallet invoca con successo l'API Key Attestation con il `client_data_hash` e ottiene un attestato firmato dalla Key Attestation API.
WP_134	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Richiesta Key Binding	Istanza del Wallet invia una richiesta Key Binding che contiene un parametro `assertion` il cui valore è un JWT firmato.
WP_134a	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Firma JWT Key Binding	Istanza del Wallet firma con successo il JWT richiesta Key Binding con la chiave pubblica Wallet Hardware.
WP_135	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Validazione Richiesta di Inizializzazione Wallet	Il Fornitore del Wallet controlla con successo i parametri della Richiesta di Inizializzazione: il `nonce` e `key_attestation`. Se validi, restituisce una risposta di conferma.
WP_135a	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Verifica unicità nonce	Il Fornitore del Wallet rifiuta qualsiasi Richiesta di Inizializzazione contenente un nonce che non ha generato o che è già stato utilizzato.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_135b	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Validazione Key Attestation per linee guida	Il Fornitore del Wallet valida il `key_attestation` usando le linee guida produttore e conferma che il dispositivo soddisfa i suoi requisiti minimi sicurezza.
WP_136	Inizializzazione / Registrazione Wallet, Ciclo di vita, Sicurezza	Verifica associazione crittografica	Il Fornitore del Wallet verifica con successo l'associazione crittografica tra il `hardware_key_tag`, `hardware_key_pub`, il nonce, e il `client_data_hash` fornito nel parametro `key_attestation`.
WP_137	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Registrazione Istanza del Wallet	Il Fornitore del Wallet registra con successo Istanza del Wallet, archiviando il suo `hardware_key_tag`, `hardware_key_pub`, e altre informazioni dispositivo.
WP_138	Inizializzazione / Attivazione Wallet, Ciclo di vita, Sicurezza	Creazione account utente	Il Fornitore del Wallet crea un account Utente associato al `hardware_key_tag` dell'Istanza del Wallet.
WP_139	Disinstallazione Wallet, Ciclo di vita, Sicurezza	Eliminazione chiave/stato su disinstallazione	Istanza del Wallet si disinstalla e rimuove tutte le chiavi locali e dati applicazione.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_140	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Richiesta di Emissione della Wallet Instance Attestation (WIA)	Istanza del Wallet costruisce con successo il JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) con i claim richiesti: `integrity_assertion`, `hardware_signature`, `nonce`, `hardware_key_tag`, `cnf`, e altri parametri correlati alla configurazione.
WP_140a	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Controllo esistenza chiave hardware/ri-inizializzazione	Istanza del Wallet controlla l'esistenza di Cryptographic Hardware Keys; se non vengono trovate, attiva il processo ri-inizializzazione.
WP_140b	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Interoperabilità	Richiede nonce per Richiesta di Emissione della Wallet Instance Attestation (WIA)	Istanza del Wallet richiede con successo e riceve un nuovo nonce dal Nonce Endpoint del Fornitore del Wallet, prima di richiedere la Wallet Instance Attestation (WIA).
WP_140c	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Interoperabilità	Calcola hash per Richiesta di Emissione della Wallet Instance Attestation (WIA)	Istanza del Wallet calcola un digest SHA-256 (`client_data_hash`) del nonce e del thumbprint del JWK `ephemeral_key_pub`.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_140d	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Interoperabilità	Firma hash con chiave privata hardware	Istanza del Wallet firma il `client_data_hash` con la chiave privata hardware e produce un `hardware_signature` valido.
WP_140e	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Interoperabilità	Ottiene Integrity Assertion firmato dal Servizio di Integrità del Dispositivo.	Istanza del Wallet richiede con e riceve con successo una `integrity_assertion` firmata dal Servizio di Integrità del Dispositivo che è associato al `client_data_hash`.
WP_140f	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Includere claim `cnf`	Il claim `cnf` nel payload del JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) contiene la chiave pubblica effimera, collegando la chiave all'attestato.
WP_141	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Firma della Richiesta di Emissione della Wallet Instance Attestation (WIA)	Istanza del Wallet firma il JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) con la chiave privata effimera.
WP_142	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Invio del JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) al Fornitore del Wallet	Istanza del Wallet invia il JWT richiesta firmato come parametro `assertion` all'Endpoint di Issuance degli Attestati Elettronici del Fornitore del Wallet.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_143	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica JWT Richiesta Attestato	Il Fornitore del Wallet esegue con successo una validazione completa della Richiesta di Emissione della Wallet Instance Attestation (WIA), inclusa la sua firma, i suoi parametri, e le verifiche crittografiche associate.
WP_143a	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Validazione header HTTP per Richiesta di Emissione della Wallet Instance Attestation (WIA)	Il Fornitore del Wallet valida con successo l'header del JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) per contenere parametri `alg`, `kid`, e `typ` validi.
WP_143b	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica firma JWT in Richiesta di Emissione della Wallet Instance Attestation (WIA)	Il Fornitore del Wallet verifica con successo la firma del JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) utilizzando la chiave pubblica nel JWK fornito.
WP_143c	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica unicità nonce per Wallet Instance Attestation (WIA)	Il Fornitore del Wallet rifiuta il JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) se il nonce non è stato generato da esso stesso o è stato precedentemente utilizzato.
WP_143d	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica Istanza del Wallet registrata	Il Fornitore del Wallet conferma che la Richiesta di Emissione della Wallet Instance Attestation (WIA) provenga da un'Istanza del Wallet valida e attualmente registrata; se non rifiuta la richiesta.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_143e	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Interoperabilità	Validazione firma hardware	Il Fornitore del Wallet ricostruisce con successo i client_data, e valida l'`hardware_signature` utilizzando la chiave pubblica Hardware dell'Istanza del Wallet registrata precedentemente.
WP_143f	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Validazione Integrity Assertion per linee guida	Il Fornitore del Wallet valida con successo l'`integrity_assertion` secondo le linee guida del produttore dispositivo.
WP_143g	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Verifica parametro `iss`	Il Fornitore del Wallet verifica che il parametro `iss` nel JWT Richiesta di Emissione della Wallet Instance Attestation (WIA) corrisponda al suo identificativo URL.
WP_144	Rilascio Wallet Instance Attestation (WIA), Ciclo di vita, Sicurezza	Rilascio Attestato	Dopo un validazione riuscita della Richiesta di Emissione della Wallet Instance Attestation (WIA), il Fornitore del Wallet rilascia una Wallet Instance Attestation (WIA) che soddisfa *la durata della Wallet Instance Attestation* (pagina 290).
WP_145	Revoca Wallet, Ciclo di vita, Interoperabilità	Recupero stato Istanza del Wallet	Istanza del Wallet invia una Richiesta di Recupero Istanze del Wallet all'Endpoint di Gestione dell'Istanza del Wallet del Fornitore del Wallet.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_146	Revoca Wallet, Ciclo di vita, Interoperabilità	Risposta stato Istanza del Wallet	Il Fornitore del Wallet valida la richiesta e restituisce all'Istanza del Wallet un elenco delle Istanze del Wallet collegate dell'Utente autenticato, inclusi i campi `ID`, `status`, e `issued_at` di ogni istanza.
WP_147	Revoca Wallet, Ciclo di vita, Interoperabilità	Corpo Richiesta di Revoca Istanza del Wallet	La Richiesta di Revoca Istanza del Wallet include nel body un oggetto con chiave `status` e valore `REVOKED`.
WP_148	Revoca Wallet, Ciclo di vita, Interoperabilità	Risposta Revoca Istanza del Wallet	Il Fornitore del Wallet aggiorna lo stato dell'Istanza del Wallet a `REVOKED` e restituisce una conferma.
WP_149	Revoca Wallet, Ciclo di vita, Sicurezza	Cancella chiavi su revoca	Nella revoca, Istanza del Wallet rimuove tutte le Cryptographic Hardware Key associate dal dispositivo all'Istanza stessa.
WP_150	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento inizializzazione (nonce non valido)	Quando una Richiesta di Inizializzazione Istanza del Wallet contiene un nonce non valido, scaduto, o già utilizzato, il Fornitore del Wallet restituisce una risposta `403 Forbidden` con codice errore `invalid_request`.
WP_151	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento inizializzazione (firma Key Attestation)	Quando una Richiesta di Inizializzazione Istanza del Wallet contiene una firma Key Attestation non valida, il Fornitore del Wallet restituisce una risposta `403 Forbidden` con codice errore `invalid_request`.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_152	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento Key Binding (Istanza del Wallet non trovata)	Quando una richiesta Key Binding riferisce un'Istanza del Wallet che non viene trovata, il Fornitore del Wallet restituisce una risposta `404 Not Found` con codice errore `not_found`.
WP_153	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento Key Binding (Istanza del Wallet revocata)	Quando una richiesta Key Binding è fatta per un'Istanza del Wallet che è stata revocata, il Fornitore del Wallet restituisce una risposta `403 Forbidden` con codice errore `invalid_request`.
WP_154	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento Key Binding (PoP non valido)	Quando una richiesta Key Binding contiene una Proof of Possession non valida (`hardware_signature`), il Fornitore del Wallet restituisce una risposta `403 Forbidden` con codice errore `invalid_request`.
WP_155	Inizializzazione / Registrazione Wallet, Ciclo di vita, Interoperabilità	Fallimento Key Binding (fallimento Integrity Assertion)	Quando l'Integrity Assertion in una richiesta Key Binding non passa la validazione (es. il dispositivo è manomesso), il Fornitore del Wallet restituisce una risposta `403 Forbidden` con codice errore `invalid_request`.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_156	Trust, Sicurezza	WUA distinta dalla WIA	La Wallet Unit Attestation (WUA) è emessa e verificata come manufatto distinto dalla WIA. Le chiavi private associate a una WUA sono memorizzate nello stesso Keystore o WSCD delle chiavi della Credenziale Digitale che la WUA attesta. I test NON DEVONO usare il nome legacy «Wallet Attestation» per nessuno dei due manufatti.
WP_157	Issuance, Autenticazione	Autenticazione per l'emissione di IT-Wallet ID	L'Istanza del Wallet ottiene un IT-Wallet ID usando CieID LoA High come metodo primario, oppure il flusso eID Substantial Authentication with MRTD Verification come alternativa. Il flusso è distinto dall'emissione di un PID High.
WP_158	Issuance, Autenticazione	`authorization_details` L2+	Quando si richiede un IT-Wallet ID con eID Substantial Authentication with MRTD Verification, il Request Object PAR include un oggetto `authorization_details` con `type` uguale a `it_l2+document_proof` e i claim previsti da *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482). Se quell'oggetto è assente, l'EAA Provider autentica l'Utente con CieID LoA High.
WP_159	Issuance, Autenticazione, Sicurezza	Binding identità IdP–MRTD in L2+	Dopo l'autenticazione primaria (SPID o CieID Substantial), l'EAA Provider correla l'identità dell'asserzione IdP con l'identità nell'MRTD (CIE). Il confronto usa almeno il codice fiscale o un identificativo univoco equivalente presente in entrambe le fonti. Una discordanza interrompe l'emissione. Se l'autenticazione primaria è LoA High, la Phase 3 (MRTD PoP) è saltata.
WP_160	Presentation, Sicurezza	Presentazione offline e collocazione delle chiavi	La presentazione offline o in prossimità di una Credenziale Digitale ha successo solo se le chiavi private associate sono in un Keystore o WSCD locale. Le credenziali le cui chiavi sono in un WSCD remoto non sono presentate offline.

continues on next page

Tabella 20.32 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultati Attesi
WP_161	Trust, Sicurezza	Endpoint PDND del Fornitore del Wallet autenticati	Le chiamate al Catalogo e-Service PDND del Fornitore del Wallet, incluso Notify User Death, sono autenticate e autorizzate. Le richieste non autenticate sono rifiutate. Un Notify User Death riuscito porta alla revoca dell'Istanza del Wallet come per le altre revoche avviate dal PID/IT-Wallet ID Provider.

Matrice dei Test per il Credential Issuer

Questa sezione fornisce l'insieme dei test progettati per implementatori tecnici e team di sviluppo responsabili della creazione e del deployment di soluzioni Credential Issuer. È anche destinata agli organismi di valutazione che ispezionano e validano le implementazioni di soluzioni Credential Issuer.

ID	Scopo	Descrizione	Risultato Atteso
CI_001	Trust, Sicurezza	Pubblicazione della Entity Configuration	L'entità di federazione pubblica la propria Entity Configuration nell'endpoint . well-known/openid-federation.
CI_002	Trust, Interoperabilità	Verifica del media type della Entity Configuration	La risposta HTTP della Entity Configuration è impostata sul media type application/entity-statement+jwt.
CI_003	Trust, Sicurezza	Crittografia della Entity Configuration	La Entity Configuration è firmata crittograficamente.
CI_004	Trust, Sicurezza	Inclusione della Chiave Pubblica nella Entity Configuration e nella Subordinate Statement	La Entity Configuration e la Subordinate Statement emessa dall'entità superiore immediata includono entrambe la parte pubblica della chiave.
CI_005	Trust, Sicurezza	Trust Mark nella Entity Configuration	La Entity Configuration contiene uno o più Trust Mark.
CI_006	Trust, Interoperabilità	Parametri delle Entity Configuration	Le Entity Configuration hanno in comune i seguenti parametri: iss, sub, iat, exp, jwks, metadata.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_007	Annullato	Annullato	Annullato
CI_008	Trust, Interoperabilità	Metadata del Credential Issuer	Il Credential Issuer fornisce correttamente i seguenti tipi di metadata: *federation_entity*, *oauth_authorization_server* e *openid_credential_issuer*.
CI_009	Trust, Interoperabilità	Inclusione dei Metadata *openid_credential_verifier* nell'Autenticazione Utente tramite Wallet	Quando i Fornitori di Attestati Elettronici di Attributi (Qualificati) autenticano gli utenti tramite la loro Istanza Wallet, i metadata *openid_credential_verifier* sono inclusi oltre ai parametri metadata obbligatori.
CI_010	Emissione, Interoperabilità	Struttura della URI della Credential Offer	Il Credential Issuer genera una Credential Offer composta da un singolo parametro URI di query chiamato "credential_offer", garantendo che l'URL sia ben formato e contenga solo questo parametro per i dati dell'offerta.
CI_011	Emissione, Interoperabilità	Metodi di Consegna della Credential Offer	L'URL della Credential Offer è incorporato con successo in codici QR o incluso come pulsante cliccabile href in pagine HTML.
CI_012	Emissione, Interoperabilità	Parametri Obbligatori della Credential Offer	La Credential Offer contiene tutti e tre i parametri obbligatori (credential_issuer, credential_configuration_ids e grants) con valori validi.
CI_013	Emissione, Interoperabilità	Struttura del parametro Grants nella Credential Offer	Il parametro grants contiene correttamente un oggetto `authorization_code` che include entrambi i sotto-parametri obbligatori (`issuer_state` e `authorization_server`) con valori appropriati.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_014	Emissione, Interoperabilità	Compilazione dell'Oggetto Credential	Il Credential Issuer garantisce che l'Oggetto Credential sia compilato correttamente con tutti i campi richiesti, formattazione corretta e strutture dati valide prima della consegna.
CI_015	Emissione, Sicurezza	Validazione della Firma del Request Object nella PAR	Il Credential Issuer valida correttamente la firma del Request Object.
CI_015a	Emissione, Sicurezza	Elaborazione dell'algoritmo nell'header del Request Object della PAR	Il Credential Issuer utilizza l'algoritmo specificato nel parametro di header alg (**RFC 9126**/**RFC 9101**) per validare la firma del Request Object.
CI_015b	Emissione, Sicurezza	Recupero della Chiave Pubblica dalla Wallet Instance Attestation (WIA) nella PAR	Il Credential Issuer recupera correttamente la chiave pubblica dal claim cnf. jwk della Wallet Instance Attestation (WIA).
CI_015c	Emissione, Sicurezza	Riferimento al Key Identifier JWT nella PAR	Il Credential Issuer fa correttamente riferimento alla chiave giusta tramite il parametro kid nell'header JWT.
CI_015d	Emissione, Sicurezza	Validazione dell'Integrità della Firma Crittografica nella PAR	Il Credential Issuer completa correttamente la validazione dell'integrità della firma crittografica prima di procedere con ulteriori verifiche.
CI_016	Emissione, Interoperabilità	Codifica dei Parametri HTTP POST nella PAR	Il Credential Issuer elabora correttamente le richieste HTTP POST con parametri nel corpo del messaggio codificati in formato application/ x-www-form-urlencoded.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_017	Emissione, Interoperabilità	Interpretazione di Scope e Authorization Details nella PAR	La Pushed Authorization Request contiene sia il valore scope sia il parametro authorization_details. Il Credential Issuer elabora ciascun parametro in modo indipendente.
CI_017a	Emissione, Interoperabilità	Authorization Details nella PAR	Quando sia scope che authorization_details richiedono lo stesso tipo di Credential, il Credential Issuer segue le specifiche fornite dall'oggetto authorization_details.
CI_018	Annullato	Annullato	Annullato
CI_019	Emissione, Sicurezza	Verifica di Conformità dell'Algoritmo nella PAR	Il Credential Issuer verifica che l'algoritmo di firma nell'header alg corrisponda a uno degli algoritmi approvati nella sezione Algoritmi Crittografici; rifiuta richieste con algoritmi non conformi e restituisce un errore appropriato.
CI_020	Emissione, Sicurezza	Validazione di Consistenza del Client ID nella PAR	Il Credential Issuer conferma che il client_id nel corpo della richiesta PAR corrisponda esattamente al claim client_id nel Request Object; valori non corrispondenti causano il rifiuto con messaggio di errore chiaro.
CI_021	Emissione, Sicurezza	Corrispondenza Issuer-Client ID nella richiesta PAR	Il Credential Issuer valida che il claim iss nel Request Object corrisponda al claim client_id nello stesso Request Object (**RFC 9126**/**RFC 9101**); discrepanze portano al rifiuto della richiesta.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_022	Emissione, Sicurezza	Verifica dell'Audience Claim nella PAR	Il Credential Issuer conferma che il claim aud nel Request Object sia uguale al proprio identificativo (**RFC 9126**/**RFC 9101**); valori errati comportano il rifiuto immediato.
CI_023	Emissione, Sicurezza	Rifiuto del parametro Request URI nella PAR	Il Credential Issuer rileva e rifiuta qualsiasi richiesta PAR contenente il parametro request_uri (**RFC 9126**), restituendo un errore che segnala il parametro non supportato.
CI_024	Emissione, Sicurezza	Validazione dei Parametri Obbligatori nella PAR	Il Credential Issuer verifica la presenza di tutti i parametri HTTP obbligatori nel Request Object e valida i loro valori rispetto alle specifiche di tabella definite (**RFC 9126**); parametri mancanti o non validi causano risposte di errore strutturate.
CI_025	Emissione, Sicurezza	Controllo della Scadenza del Token nella PAR	Il Credential Issuer verifica che il Request Object non sia scaduto controllando il claim exp rispetto al tempo corrente; i token scaduti sono rifiutati con messaggi di errore legati al timestamp.
CI_026	Emissione, Sicurezza	Validazione del Tempo di Emissione del Token nella PAR	Il Credential Issuer conferma che il claim iat rappresenti un timestamp passato.
CI_026a	Emissione, Sicurezza	Rifiuto Consigliato per Tempo di Emissione Token nella PAR	Il Credential Issuer rifiuta richieste in cui iat è superiore a più di 5 minuti dal tempo corrente (**RFC 9126**); violazioni temporali generano errori "invalid_request".

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_027	Emissione, Sicurezza	Prevenzione di Replay Attack nella PAR	Il Credential Issuer verifica che il claim jti nel Request Object non sia stato usato prima dall'Istanza Wallet identificata dal client_id.
CI_028	Emissione, Sicurezza, Interoperabilità	Validazione OAuth Client Attestation PoP	Il Credential Issuer valida con successo il parametro OAuth-Client-Attestation-PoP secondo la Sezione 5 di [OAUTH-ATTESTATION-CLIENT-AUTH], confermando la prova di possesso e rifiutando attestazioni non valide con risposte di errore dettagliate.
CI_029	Emissione, Fiducia	Verifica dell'affidabilità dell'istanza del Wallet	Il Credential Issuer verifica con successo l'affidabilità e l'appartenenza indiretta alla Federazione dell'istanza del Wallet attraverso una validazione completa della Wallet Instance Attestation (WIA).
CI_030	Emissione, Fiducia	Validazione dell'appartenenza alla Federazione del Wallet Provider	Il Credential Issuer conferma che il Wallet Provider (emittente della Wallet Instance Attestation (WIA)) è un membro della Federazione riconosciuto e affidabile controllando i registri e le liste di fiducia della Federazione.
CI_031	Emissione, Sicurezza	Validazione della firma crittografica della Wallet Instance Attestation (WIA)	Il Credential Issuer valida con successo la firma crittografica della Wallet Instance Attestation (WIA) utilizzando la chiave pubblica del Wallet Provider, garantendo integrità e autenticità della firma.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_032	Emissione, Sicurezza	Controllo della scadenza della Wallet Instance Attestation (WIA)	Il Credential Issuer verifica che la Wallet Instance Attestation (WIA) non sia scaduta al momento della verifica confrontando i timestamp dichiarati con l'orario corrente.
CI_033	Emissione, Sicurezza	Accettazione delle chiavi crittografiche attestate nella Wallet Instance Attestation (WIA)	Il Credential Issuer accetta e utilizza solo chiavi crittografiche correttamente derivate dall'istanza del Wallet attestata durante il processo di emissione della credenziale.
CI_034	Emissione, Sicurezza	Verifica della sicurezza e conformità del dispositivo tramite Wallet Instance Attestation (WIA)	Il Credential Issuer si affida alle dichiarazioni della Wallet Instance Attestation (WIA) per confermare che l'istanza del Wallet operi su un dispositivo sicuro e affidabile che soddisfi gli standard di sicurezza richiesti dall'Issuer.
CI_035	Emissione, Fiducia	Valutazione della catena di fiducia del Wallet Provider	Il Credential Issuer valuta con successo l'intera catena di fiducia dell'emittente della Wallet Instance Attestation (WIA) (Wallet Provider).
CI_036	Emissione, Fiducia, Interoperabilità	Recupero dei metadati della Federazione	Il Credential Issuer utilizza con successo gli endpoint API della Federazione (. well-known/openid-federation, /fetch) per recuperare i metadati e le configurazioni aggiornate dei partecipanti alla Federazione, inclusi i Wallet Provider.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_037	Emissione, Fiducia, Interoperabilità	Stabilimento della fiducia nel Wallet Provider	Il Credential Issuer stabilisce la fiducia nel Wallet Provider come entità autorizzata della Federazione interrogando gli endpoint API della Federazione (ad es. .well-known/openid-federation, / fetch) e validando lo stato federativo del provider tramite canali ufficiali e processi di verifica della fiducia.
CI_038	Annullato	Annullato	Annullato
CI_039	Annullato	Annullato	Annullato
CI_040	Emissione, Sicurezza	Durata consigliata di validità del request_uri nella risposta PAR	Il tempo di validità del request_uri è impostato a meno di un minuto.
CI_041	Emissione, Sicurezza	Integrazione di un valore casuale crittografico nella risposta PAR	Il request_uri generato include un valore casuale crittografico di almeno 128 bit.
CI_042	Emissione, Sicurezza	Limitazione consigliata della lunghezza del request_uri nella risposta PAR	Il request_uri completo non supera i 512 caratteri ASCII.
CI_043	Emissione, Interoperabilità	Risposta di verifica positiva della risposta PAR	Quando la verifica ha successo, il Credential Issuer restituisce una risposta HTTP con codice di stato 201.
CI_044	Emissione, Interoperabilità	Struttura JSON della risposta PAR	Il corpo del messaggio di risposta HTTP utilizza il tipo media application/json (**RFC 8259**) e include i parametri richiesti di livello superiore.
CI_044a	Emissione, Sicurezza	Parametro request_uri nella risposta PAR	La risposta HTTP include il parametro request_uri contenente l'URI di autorizzazione generato e utilizzabile una sola volta.
CI_044b	Emissione, Sicurezza	Parametro expires_in nella risposta PAR	La risposta HTTP include il parametro expires_in che specifica la durata di validità in secondi.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_045	Emissione, Interoperabilità	Tabella dei codici di stato HTTP per la risposta PAR	Quando l'elaborazione della richiesta PAR incontra errori, il Credential Issuer risponde come definito in **RFC 9126**, in accordo con la *Tabella dei Codici di Stato HTTP* (pagina 601).
CI_046	Emissione, Sicurezza e Privacy	Verifica dell'identità dell'utente durante la richiesta di autorizzazione	L'Authorization Server verifica con successo l'identità dell'utente proprietario della credenziale tramite meccanismi di autenticazione appropriati, confermando la titolarità prima di procedere con l'autorizzazione della credenziale.
CI_047	Emissione, Sicurezza	Monouso e scadenza del request_uri nella richiesta di autorizzazione	Il Credential Issuer tratta ogni valore request_uri come a utilizzo singolo e rifiuta con successo tutte le richieste scadute.
CI_048	Emissione, Sicurezza	Tolleranza opzionale per richieste duplicate nella richiesta di autorizzazione	Il Credential Issuer consente richieste duplicate quando derivano da ricaricamento o aggiornamento del user-agent da parte dell'utente (derivato da **RFC 9126**).
CI_049	Emissione, Sicurezza	Identificazione della richiesta PAR nella richiesta di autorizzazione	Il Credential Issuer identifica e correla con successo ogni richiesta di autorizzazione come risultato diretto di una PAR precedentemente inviata (derivato da **RFC 9126**).
CI_050	Emissione, Sicurezza	Obbligatorietà del parametro request_uri nella richiesta di autorizzazione	Il Credential Issuer rifiuta tutte le richieste di autorizzazione che non contengono il parametro request_uri, poiché la PAR è l'unico metodo per trasmettere richieste di autorizzazione dall'istanza del Wallet (derivato da **RFC 9126**).

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_051	Emissione, Sicurezza	Autenticazione CieID High per il PID	Il PID Provider esegue con successo l'autenticazione dell'utente basata sullo schema CieID con LoA High (CIE L3). Questo caso si applica al PID, non all'IT-Wallet ID.
CI_052	Emissione, Sicurezza e Privacy	Consenso dell'utente per l'emissione del PID	Il PID Provider ottiene il consenso esplicito dell'utente prima di procedere con l'emissione del PID.
CI_053	Emissione, Privacy	Raccolta opzionale dei dati di contatto	Il PID Provider PUÒ richiedere i dati di contatto dell'utente (email, numero di telefono) per l'invio di notifiche relative al PID emesso.
CI_054	Presentazione, Sicurezza di emissione	Autenticazione utente basata su PID o IT-Wallet ID	Il (Q)EAA Provider esegue con successo l'autenticazione dell'utente richiedendo e validando un PID valido dall'istanza del Wallet, oppure un IT-Wallet ID valido secondo le proprie politiche di autenticazione, come definito in credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.
CI_055	Presentazione, Emissione, Interoperabilità	Utilizzo del protocollo Open ID4VP	Il (Q)EAA Provider utilizza il protocollo OpenID4VP per richiedere la presentazione del PID dall'istanza del Wallet, oppure dell'IT-Wallet ID secondo le proprie politiche di autenticazione, come definito in credential-issuance-endpoint:Selezione del Metodo di Autenticazione dell'Utente.
CI_056	Presentazione, Emissione, Sicurezza	Consegna della richiesta di presentazione	Il (Q)EAA Provider fornisce con successo la richiesta di presentazione al Wallet.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_057	Emissione, Privacy	Raccolta opzionale dei dati di contatto per le credenziali	I Credential Issuer richiedono i dati di contatto dell'utente (email, numero di telefono) per l'invio di notifiche relative agli Attestati Elettronici emessi.
CI_058	Emissione, Interoperabilità	Validazione dei parametri della risposta di autorizzazione	Il Credential Issuer invia con successo una risposta con codice di autorizzazione all'istanza del Wallet che include tutti e tre i parametri richiesti.
CI_058a	Emissione, Sicurezza	Validazione del parametro code nella risposta di autorizzazione	La risposta con codice di autorizzazione include il parametro code.
CI_058b	Emissione, Sicurezza	Validazione del parametro state nella risposta di autorizzazione	La risposta con codice di autorizzazione include il parametro state corrispondente alla richiesta originale.
CI_058c	Emissione, Sicurezza	Validazione del parametro iss nella risposta di autorizzazione	La risposta con codice di autorizzazione include il parametro iss che identifica l'issuer.
CI_059	Emissione, Interoperabilità	Tabella dei codici di stato HTTP della risposta di autorizzazione	Quando la risposta di autorizzazione incontra errori, l'Authorization Server reindirizza l'utente alla redirect_uri con codice di stato HTTP 302 secondo la tabella dei codici di stato HTTP.
CI_060	Emissione, Sicurezza	Verifica dell'emissione del codice di autorizzazione nella richiesta di token	Il Credential Issuer garantisce che il codice di autorizzazione sia emesso all'istanza del Wallet autenticata (**RFC 6749**) e che non sia stato riutilizzato.
CI_061	Emissione, Sicurezza	Verifica di validità e utilizzo del Codice di Autorizzazione nella Richiesta di Token	Il Credential Issuer verifica che il codice di autorizzazione sia valido e non sia stato precedentemente utilizzato (**RFC 6749**).

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_061a	Emissione, Sicurezza	Verifica del `code_verifier` PKCE nella Richiesta di Token	Il Credential Issuer verifica che il parametro `code_verifier` sia presente e corrisponda al `code_challenge` associato al codice di autorizzazione (**RFC 7636**); in caso di mancata corrispondenza, la richiesta viene rifiutata.
CI_062	Emissione, Sicurezza	Validazione della corrispondenza del Redirect URI nella Richiesta di Token	Il Credential Issuer conferma che il redirect_uri corrisponda esattamente al valore incluso nel precedente Request Object (vedi Sezione 3.1.3.1 di [OIDC]).
CI_063	Emissione, Sicurezza	Validazione del DPoP Proof JWT nella Richiesta di Token	Il Credential Issuer valida correttamente il DPoP Proof JWT secondo la Sezione 4.3 di (**RFC 9449**).
CI_064	Emissione, Interoperabilità	Fornitura dell'Access Token nella risposta di Token	Il Credential Issuer fornisce al Wallet Instance un Access Token valido dopo un'autorizzazione avvenuta con successo.
CI_065	Emissione, Interoperabilità	Fornitura opzionale di Refresh Token	Se supportato dal Credential Issuer, viene fornito al Wallet Instance un Refresh Token.
CI_066	Emissione, Sicurezza	Binding crittografico di Access Token e Refresh Token alla chiave DPoP	Sia l'Access Token che il Refresh Token (se emesso) sono crittograficamente vincolati alla chiave DPoP.
CI_067	Emissione, Interoperabilità	Tabella dei Codici di Stato HTTP per le risposte di Token	In caso di errori durante la validazione della Richiesta di Token, l'Authorization Server restituisce una risposta di errore come definito in **RFC 6749**, in accordo con la Tabella dei Codici di Stato HTTP.
CI_068	Emissione, Interoperabilità	Fornitura di c_nonce	Il Credential Issuer fornisce un valore c_nonce al Wallet Instance.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_069	Emissione, Sicurezza	Formato e Sicurezza del c_nonce	Il parametro c_nonce è fornito come stringa con sufficiente imprevedibilità per prevenire attacchi di guessing e funge da sfida crittografica che il Wallet Instance utilizza per creare la prova di possesso della chiave (proofs claim).
CI_070	Emissione, Sicurezza	Riutilizzo e rinnovo del c_nonce	Il valore c_nonce ricevuto può essere riutilizzato dal Wallet Instance per creare più proof finché il Credential Issuer non ne fornisce uno nuovo.
CI_071	Emissione, Interoperabilità	Validazione delle Claim obbligatorie nel JWT Proof	Il JWT proof include correttamente tutte le claim obbligatorie come specificato nella tabella della Richiesta di Token.
CI_072	Emissione, Sicurezza	Validazione delle Claim obbligatorie nel JWT Proof per emissione in batch	Il Credential Issuer verifica correttamente che l'attributo jwk in ogni key proof sia univoco.
CI_073	Emissione, Interoperabilità	Dichiarazione del tipo di Key Proof nella Richiesta di Credenziale	Il key proof contiene i parametri di header appropriati definiti per il rispettivo tipo di proof.
CI_074	Emissione, Sicurezza	Obbligo di algoritmo asimmetrico nella Richiesta di Credenziale	Il parametro di header alg indica un algoritmo di firma digitale asimmetrico registrato e non è mai impostato su none.
CI_075	Emissione, Sicurezza	Verifica della firma della chiave pubblica nella Richiesta di Credenziale	La firma sul key proof viene verificata correttamente utilizzando la chiave pubblica specificata nel parametro di header.
CI_076	Emissione, Sicurezza	Esclusione della chiave privata negli header della Richiesta di Credenziale	I parametri di header non contengono alcun materiale di chiave privata.
CI_077	Emissione, Sicurezza	Corrispondenza del c_nonce nella Richiesta di Credenziale	Quando un valore c_nonce è stato precedentemente fornito dal server, la claim nonce nel JWT corrisponde esattamente a questo valore.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_078	Emissione, Sicurezza	Validità temporale del JWT nella Richiesta di Credenziale	Il tempo di creazione del JWT (tramite claim iat o timestamp gestito dal server attraverso la claim nonce) rientra nella finestra temporale accettabile dal server.
CI_079	Annullato	Annullato	Annullato
CI_080	Emissione, Interoperabilità	Generazione raccomandata di nuove chiavi crittografiche nella Richiesta di Credenziale	È raccomandato che vengano generate nuove chiavi crittografiche per la Richiesta di Credenziale.
CI_081	Emissione, Sicurezza	Supporto raccomandato al Deferred Flow	Quando la Credenziale richiesta non può essere emessa immediatamente e richiede più tempo, il Credential Issuer supporta il Deferred Flow.
CI_081a	Emissione, Sicurezza	Coerenza nell'emissione batch con Deferred Flow	Quando si utilizza il Deferred Flow per l'emissione batch, lo stesso transaction_id consente il recupero di tutte le Credenziali richieste nel batch.
CI_082	Emissione, Sicurezza	Validazione del DPoP JWT Proof e dell'Access Token nella Risposta di Credenziale	Il Credential Issuer valida correttamente il DPoP JWT Proof secondo le procedure della Sezione 4.3 di (**RFC 9449**) e conferma che l'Access Token sia valido e idoneo per la Credenziale richiesta.
CI_083	Emissione, Sicurezza	Validazione della prova di possesso del materiale crittografico nella Risposta di Credenziale	Il Credential Issuer valida la prova di possesso della chiave alla quale sarà vincolata la nuova Credenziale, secondo l'Appendice F.4 di Open ID4VCI.
CI_084	Emissione, Sicurezza	Creazione e Binding della Credenziale nella Risposta	Quando tutte le validazioni hanno esito positivo, il Credential Issuer crea una nuova Credenziale crittograficamente vincolata al materiale di chiave validato e la fornisce al Wallet Instance.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_084a	Emissione, Sicurezza	Controllo del tipo di Credenziale	Il Credential Issuer garantisce che il tipo di Credenziale corrisponda a quello richiesto prima dell'emissione.
CI_085	Emissione, Interoperabilità	Tabella dei Codici di Stato HTTP per la Risposta di Credenziale	Quando la Richiesta di Credenziale non contiene un Access Token valido, il Credential Endpoint restituisce una risposta di errore come definito nella Sezione 3 di [**RFC 6750**], in accordo con la *Tabella dei Codici di Stato HTTP* (pagina 618).
CI_086	Emissione, Interoperabilità	Notification ID unificato per operazioni batch	Per gli Attestati Elettronici emessi in batch, un unico notification_id copre l'intero set emesso. La risposta di notifica si applica a tutti gli Attestati: qualsiasi fallimento parziale è trattato come fallimento dell'intero batch.
CI_087	Emissione, Interoperabilità	Tabella dei Codici di Stato HTTP per la Risposta di Notifica	Quando la Richiesta di Notifica non contiene un Access Token valido, il Notification Endpoint restituisce una risposta di errore come definito nella Sezione 3 di [**RFC 6750**], in accordo con la *Tabella dei Codici di Stato HTTP* (pagina 623).
CI_088	Emissione, Sicurezza	Restrizione dello scope dell'Access Token	L'Access Token ottenuto tramite Refresh Token è limitato a tre endpoint specifici: Deferred endpoint, Notification endpoint e Credential endpoint.
CI_088a	Emissione, Sicurezza	Autorizzazione di accesso al Deferred Endpoint	L'Access Token consente l'accesso al Deferred endpoint per ottenere nuovi Attestati Elettronici dopo il interval o la notifica di readiness.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_088b	Emissione, Sicurezza	Autorizzazione di accesso al Notification Endpoint	L'Access Token consente l'accesso al Notification endpoint per notificare l'eliminazione di un Attestato Elettronico al Credential Issuer.
CI_089c	Emissione, Sicurezza	Autorizzazione di accesso al Credential Endpoint	L'Access Token consente l'accesso al Credential endpoint per il rinnovo/ re-emissione di Attestati Elettronici esistenti.
CI_090	Annullato	Annullato	Annullato
CI_091	Emissione, Interoperabilità	Validazione dell'OAuth Client Attestation PoP per il Refresh	Il Credential Issuer valida correttamente il parametro OAuth-Client-Attestation-PoP secondo la Sezione 5 di [OAUTH-ATTESTATION-CLIENT-AUTH].
CI_092	Emissione, Sicurezza	Validazione del DPoP Proof JWT per il Refresh	Il Credential Issuer valida il DPoP Proof JWT secondo la Sezione 4.3 di (**RFC 9449**).
CI_093	Emissione, Sicurezza	Controllo di validità e binding del Refresh Token	Il Credential Issuer verifica che il Refresh Token non sia scaduto, non sia stato revocato e sia vincolato alle stesse chiavi DPoP usate nel DPoP Proof JWT.
CI_094	Emissione, Sicurezza	Generazione e Binding dell'Access Token	Quando tutte le verifiche hanno esito positivo, il Credential Issuer genera un nuovo Access Token e un nuovo Refresh Token, entrambi vincolati alla chiave DPoP.
CI_095	Emissione, Sicurezza	Risposta positiva con Refresh Token	Sia l'Access Token che il Refresh Token sono inviati al Wallet Instance.
CI_096	Emissione, Sicurezza	Gestione errori per Refresh Token non valido	Quando il Refresh Token è scaduto o non valido, il Credential Issuer restituisce una risposta di errore con il tipo di errore impostato su invalid_grant.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_097	Emissione, Sicurezza	Protezione della riservatezza del Refresh Token	La riservatezza dei Refresh Token è garantita sia in transito che a riposo tramite cifratura appropriata e meccanismi sicuri di gestione.
CI_098	Emissione, Sicurezza	Trasmissione dei Refresh Token protetta da TLS	Tutte le trasmissioni di token utilizzano connessioni protette da TLS, garantendo canali di comunicazione cifrati per lo scambio di token.
CI_099	Emissione, Sicurezza	Proprietà di sicurezza del Refresh Token	I Refresh Token sono generati con valori non indovinabili e protetti da modifiche tramite meccanismi crittografici di integrità.
CI_100	Annullato	Annullato	Annullato
CI_101	Emissione, Sicurezza	Coerenza di binding della chiave DPoP tra Refresh e Access Token	Gli Access Token e i Refresh Token sono vincolati alla stessa chiave DPoP.
CI_102	Emissione, Sicurezza	Obbligo di DPoP Proof per il Refresh Token	Il DPoP Proof è richiesto per tutte le operazioni di refresh per ottenere nuovi Access Token.
CI_103	Emissione, Sicurezza	Uso coerente della chiave DPoP per il Refresh Token	La stessa chiave DPoP viene utilizzata per generare i DPoP Proof degli Access Token in tutte le Richieste di Credenziale.
CI_104	Emissione, Sicurezza	Gestione della durata di utilizzo del Refresh Token	I Credential Issuer gestiscono e limitano la durata per cui i Refresh Token possono essere utilizzati per aggiornare le Credenziali, prima di richiedere il riavvio completo del processo di emissione (OPENID4VC-HAIP).
CI_105	Emissione, Sicurezza	Allineamento raccomandato delle date di scadenza nelle Credenziali riemesse	I nuovi Attestati Elettronici ottenuti tramite il flusso di ri-emissione mantengono la stessa scadenza dell'Attestato aggiornato.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_106	Emissione, Sicurezza	Obbligo di nuova emissione dopo la scadenza	Una volta che un Attestato Elettronico è scaduto, l'Utente deve completare l'intero processo di emissione per ottenere nuovi Attestati Elettronici.
CI_107	Emissione, Sicurezza	Obbligo di coerenza dell'Issuer nella Ri-emissione	I nuovi Attestati Elettronici sono emessi esclusivamente dallo stesso Credential Issuer che ha originariamente fornito le credenziali al Wallet Instance.
CI_108	Emissione, Sicurezza	Limitazione dello scope del Refresh Token per la Ri-emissione	Gli Access Token ottenuti tramite Refresh Token non possono essere utilizzati per l'emissione di nuovi Attestati Elettronici non già presenti nel Wallet Instance (prima emissione).
CI_109	Emissione, Sicurezza	Limitazione dello scope del processo di Ri-emissione	Il processo di ri-emissione è limitato a due specifici tipi di aggiornamento: aggiornamenti tecnici del modello/formato dei dati e aggiornamenti dell'insieme di attributi dell'Utente.
CI_110	Emissione, Sicurezza	Aggiornamenti tecnici senza interazione utente	Per aggiornamenti tecnici del modello/formato dei dati, la sostituzione e l'archiviazione degli Attestati Elettronici non richiedono il coinvolgimento diretto dell'Utente.
CI_111	Emissione, Sicurezza e Privacy	Autorizzazione utente per aggiornamento attributi	Per aggiornamenti dell'insieme di attributi dell'Utente, il Wallet Instance informa l'Utente delle modifiche e richiede esplicita autorizzazione prima di archiviare il nuovo Attestato Elettronico.
CI_112	Emissione, Sicurezza	Coerenza della data di scadenza nella Ri-emissione	I nuovi Attestati Elettronici mantengono la stessa data di scadenza della versione precedente.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_113	Emissione, Privacy	Notifica opzionale out-of-band per aggiornamenti di Ri-emissione	Quando un Attestato Elettronico necessita di aggiornamento, i Credential Issuer inviano notifiche agli Utenti tramite canali di comunicazione out-of-band registrati (email, SMS, notifiche push).
CI_114	Emissione, Sicurezza	Restrizione di prima emissione per i Refresh Token	Gli Access Token ottenuti tramite Refresh Token non possono essere utilizzati per la prima emissione di Attestati Elettronici.
CI_115	Annullato	Annullato	Annullato
CI_116	Emissione, Privacy	Consenso utente per la Ri-emissione basata su attributi	Nei processi di ri-emissione attivati da modifiche degli attributi, viene richiesto il consenso dell'Utente prima dell'archiviazione della nuova Credenziale Digitale.
CI_117	Modello di Dati e ciclo di vita, Interoperabilità	Attributi Utente PID domestico	Il PID è fornito con successo con gli attributi utente definiti nella rispettiva *tabella degli Attributi PID dell'Utente* (pagina 405).
CI_118	Modello di Dati e ciclo di vita, Emissione, Interoperabilità	Formati di Credenziali (Q)EAA	Le (Q)EAA sono rilasciate a un'Istanza del Wallet in formato dati SD-JWT VC o mdoc-CBOR.
CI_119	Annullato	Annullato	Annullato
CI_120	Modello di Dati e ciclo di vita, Sicurezza	Firma della credenziale SD-JWT	La Credenziale in formato SD-JWT è firmata utilizzando la chiave privata del Fornitore di Attestati Elettronici
CI_121	Modello di Dati e ciclo di vita, Interoperabilità	Fornitura Metadati Tipo SD-JWT	L'SD-JWT è fornito con successo con Metadati Tipo completi per la Credenziale Digitale emessa, conforme alle Sezioni 6 e 6.3 delle specifiche SD-JWT-VC.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_122	Modello di Dati e ciclo di vita, Sicurezza	Validazione Struttura Payload SD-JWT	Il payload SD-JWT contiene il claim richiesto *_sd_alg* e tutti gli altri claim obbligatori come specificato nella Sezione 4.1.1 SD-JWT, con formattazione e struttura corrette.
CI_123	Modello di Dati e ciclo di vita, Sicurezza	Dichiarazione Algoritmo Hash SD-JWT	Il claim *_sd_alg* è presente e impostato su un algoritmo supportato definito nella Sezione Algoritmi Crittografici, indicando l'algoritmo hash utilizzato dal Fornitore di Attestati Elettronici per generare i digest come descritto nella Sezione 4.1.1 di SD-JWT.
CI_124	Modello di Dati e ciclo di vita, Interoperabilità	Organizzazione Divulgazione Selettiva SD-JWT	I claim non selettivamente divulgabili appaiono direttamente nel payload SD-JWT così come sono, mentre i digest dei claim selettivamente divulgabili (più eventuali esche) sono organizzati correttamente all'interno degli array *_sd* come descritto nella Sezione 4. 2.4.1.
CI_125	Modello di Dati e ciclo di vita, Interoperabilità	Verifica Integrità disclosure SD-JWT	Ogni valore digest viene convalidato correttamente rispetto alla corrispondente disclosure, le cui disclosure contengono i componenti richiesti: salt casuale, nome della claim (per gli elementi oggetto) e valore della claim.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_126	Modello di Dati e ciclo di vita, Interoperabilità	Raccomandazione su Divulgazione Selettiva Oggetti Annidati SD-JWT	In un payload SD-JWT annidato, ogni claim a ogni livello della struttura JSON è esplicitamente contrassegnato come selettivamente divulgabile o meno. Di conseguenza, il claim *_sd* contenente digest può legittimamente apparire più volte a livelli diversi all'interno dell'SD-JWT.
CI_127	Modello di Dati e ciclo di vita, Interoperabilità	Posizionamento digest Disclosure SD-JWT	Le disclosures di elementi array sono posizionate correttamente, con digest e digest esca che sostituiscono i valori claim originali nelle loro posizioni array esatte, come specificato nella Sezione 4.2.4.2 SD-JWT.
CI_128	Modello di Dati e ciclo di vita, Sicurezza	Calcolo Digest Elementi Array SD-JWT	I valori digest degli elementi array sono calcolati utilizzando una funzione hash sulle disclosures, contenenti: un salt casuale e l'elemento array.
CI_129	Modello di Dati e ciclo di vita, Privacy	SD-JWT Array Disclosures	La divulgazione selettiva a livello array opera correttamente, consentendo ai Titolari di divulgare selettivamente interi array o singole voci all'interno degli array, come definito nella Sezione 4.2.6 di SD-JWT.
CI_130	Modello di Dati e ciclo di vita, Interoperabilità	Consegna delle Disclosure nel Formato Combinato SD-JWT	Le Disclosures sono fornite con successo al Titolare insieme all'SD-JWT nel Formato Combinato per l'Emissione, formattate come una serie ordinata di valori codificati base64url con ogni valore separato correttamente da un singolo carattere tilde ('~').

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_131	Modello di Dati e ciclo di vita, Interoperabilità	Parametro Header JOSE SD-JWT	L'header JOSE contiene i parametri definiti nella *Tabella degli header SD-JWT della Credenziale* (pagina 381).
CI_132	Modello di Dati e ciclo di vita, Interoperabilità	Claim Payload SD-JWT	Il payload JWT contiene claim nella Credenziale *Tabella dei Parametri SD-JWT della Credenziale* (pagina 381)
CI_133	Modello di Dati e ciclo di vita, Interoperabilità	Struttura Parametro Lista Stato SD-JWT	Quando il parametro status è impostato su status_list, è un Oggetto JSON contenente i seguenti sotto-parametri: *idx* e *uri*
CI_134	Modello di Dati e ciclo di vita, Interoperabilità	Recupero Opzionale Metadati Tipo Credenziale SD-JWT	Il Documento JSON Metadati Tipo Credenziale è recuperato con successo direttamente dal *well-known* endpoint.
CI_134a	Modello di Dati e ciclo di vita, Interoperabilità	Corrispondenza URI Case-Insensitive SD-JWT	Quando si recuperano i Metadati Tipo Credenziale tramite vct, la corrispondenza letterale della stringa URI è eseguita in modo case-sensitive, mentre il sistema opera senza richiedere l'endpoint .well-known (come specificato nella Sezione 6.3.1 di SD-JWT-VC), mantenendo opzioni di compatibilità per implementatori che scelgono di utilizzarlo per l'interoperabilità con altri sistemi.
CI_135	Modello di Dati e ciclo di vita, Interoperabilità	Struttura Oggetto JSON Documento Metadati	Il documento *Type Metadata*, se presente, è un oggetto JSON contenente i parametri definiti nella Sezione 6.2 di SD-JWT-VC.
CI_136	Modello di Dati e ciclo di vita, Interoperabilità	Claim PID Aggiuntivi	A seconda del tipo di Credenziale Digitale specificato in vct, i dati dei claim aggiuntivi sono incorporati con successo quando richiesto

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_137	Modello di Dati e ciclo di vita, Interoperabilità	Formato Credenziale Mdoc	Gli elementi dati mdoc sono codificati con successo in formato CBOR secondo le specifiche **RFC 8949**, assicurando corretta serializzazione binaria e compatibilità con lo standard ISO/IEC 18013-5.
CI_138	Modello di Dati e ciclo di vita, Interoperabilità	Organizzazione Struttura Componenti Mdoc	La Credenziale Digitale mdoc è strutturata correttamente in componenti distinti inclusi namespace (nameSpaces) e prova crittografica (issuer Auth)
CI_139	Modello di Dati e ciclo di vita, Privacy	Verifica Integrità Oggetto Sicurezza Mobile Mdoc	L'Oggetto Sicurezza Mobile (MSO) memorizza correttamente i digest crittografici degli attributi all'interno dei nameSpaces, consentendo alle Relying Party di validare gli attributi divulgati contro i valori digest ID corrispondenti mantenendo la privacy delle informazioni non divulgate. Vedere *Mobile Security Object* (pagina 390) per dettagli
CI_140	Modello di Dati e ciclo di vita, Interoperabilità	Conformità Struttura Mdoc-CBOR	La Credenziale Digitale mdoc-CBOR è conforme con successo alla struttura richiesta come specificato nella *tabella del formato mdoc-CBOR* (pagina 389).
CI_141	Modello di Dati e ciclo di vita, Interoperabilità	Organizzazione Struttura Namespace	I nameSpaces contengono correttamente uno o più name Space, ognuno identificato correttamente da un nome univoco per la categorizzazione organizzata dei dati.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_142	Modello di Dati e ciclo di vita, Interoperabilità	Codifica IssuerSignedItem Bytes Namespaces	All'interno di ogni name Space, uno o più IssuerSigned ItemBytes sono codificati correttamente come stringhe di byte CBOR con Tag 24 (# 6.24(bstr .cbor)), apparendo come 24(«... ») in notazione diagnostica
CI_143	Modello di Dati e ciclo di vita, Interoperabilità	Attributi Informazioni Divulgazione Namespace	Ogni IssuerSignedItemBytes rappresenta con successo le informazioni di divulgazione per i digest corrispondenti all'interno dell'Oggetto Sicurezza Mobile e contiene tutti gli attributi come specificato nella *tabella degli Attributi dei Namespaces* (pagina 392)
CI_144	Modello di Dati e ciclo di vita, Interoperabilità	Inclusione ElementIdentifier Namespace	Tutti gli elementIdentifier nella *tabella* (pagina 393) dell'attributo elementIdentifiers sono inclusi correttamente nella Credenziale Digitale codificata in mdoc-CBOR all'interno dei rispettivi nameSpaces, salvo diversa specifica
CI_145	Modello di Dati e ciclo di vita, Interoperabilità	Struttura COSE Oggetto Sicurezza Mobile	L'issuerAuth rappresenta con successo l'Oggetto Sicurezza Mobile come un Documento COSE Sign1 formattato correttamente secondo **RFC 9052**, contenente la struttura dati completa richiesta con: header protetto, header non protetto, payload e componenti firma.
CI_146	Modello di Dati e ciclo di vita, Interoperabilità	Codifica Parametro Header Protetto Oggetto Sicurezza Mobile	L'header protetto contiene con successo i parametri codificati correttamente in formato CBOR secondo la *tabella* (pagina 390) corrispondente.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_147	Modello di Dati e ciclo di vita, Interoperabilità	Algoritmo Header Protetto Oggetto Sicurezza Mobile	L'header protetto contiene con successo il parametro algoritmo firma richiesto
CI_147a	Modello di Dati e ciclo di vita, Interoperabilità	Elementi Non Raccomandati nell'Header Protetto Oggetto Sicurezza Mobile	L'header protetto non contiene elementi diversi dall'algoritmo firma
CI_148	Modello di Dati e ciclo di vita, Interoperabilità	Codifica Parametro Header Non Protetto Oggetto Sicurezza Mobile	Salvo diversa specifica, l'header non protetto contiene i parametri conformemente alla *tabella* (pagina 390) corrispondente.
CI_148a	Modello di Dati e ciclo di vita, Interoperabilità	Inclusione *x5chain* Oggetto Sicurezza Mobile	La *x5chain* è inclusa correttamente nell'header non protetto
CI_149	Modello di Dati e ciclo di vita, Interoperabilità	Struttura Payload Oggetto Sicurezza Mobile	Il payload contiene con successo il Mobile SecurityObject codificato correttamente come stringa di byte (bstr) utilizzando CBOR Tag 24, con il parametro header COSE Sign content-type correttamente escluso dalla struttura.
CI_150	Modello di Dati e ciclo di vita, Interoperabilità	Conformità Attributi Oggetto Sicurezza Mobile	Il MobileSecurityObject contiene con successo tutti gli attributi richiesti come specificato nella *tabella* (pagina 391) di conformità, con formattazione e valori corretti salvo esenzione esplicita dai requisiti della specifica.
CI_151	Emissione, Interoperabilità	Verifica stato Istanza del Wallet	Il Fornitore di Attestati Elettronici verifica con successo che l'Istanza del Wallet sia in stato Operativo o Valido e procede con l'emissione della Credenziale Digitale.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_152	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Stato Credenziale per Attivazione	Una Credenziale Digitale transita con successo allo stato Valido quando la data di inizio validità è raggiunta, o quando il processo di riattivazione è completato per una (Q)EAA precedentemente sospesa.
CI_153	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Automatica Scadenza Credenziale	Una Credenziale Digitale transita con successo allo stato Scaduto quando scade automaticamente al raggiungimento della sua data di fine validità (PID/(Q)EAA EXP),
CI_154	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Processo Revoca Credenziale	Una Credenziale Digitale cambia con successo dagli stati Emesso, Valido o Sospeso allo stato Revocato quando è attivamente revocata dal Fornitore di Attestati Elettronici tramite un processo di revoca (PID/ (Q)EAA REV).
CI_155	Modello di Dati e ciclo di vita, Interoperabilità	Casi d'Uso Revoca Credenziale Digitale	La Revoca Credenziale Digitale è implementata correttamente per ogni caso d'uso
CI_155a	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Compromissione Sicurezza Tecnica	La Credenziale Digitale è revocata con successo quando viene rilevata compromissione del materiale crittografico, con invalidazione immediata e aggiornamento dello stato nei registri di revoca.
CI_155b	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Richiesta Utente	La Credenziale Digitale è revocata con successo su richiesta esplicita dell'Utente

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_155c	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Aggiornamenti Attributi	La Credenziale Digitale è revocata automaticamente quando le Fonti Autentiche notificano cambiamenti di attributi che invalidano la credenziale, innescando il processo di riemissione se applicabile.
CI_155d	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Revoca Avviata dalla Fonte	La Credenziale Digitale è revocata immediatamente quando la Fonte Autentica notifica la revoca degli attributi sottostanti
CI_155e	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Revoca Istanza del Wallet	Tutte le Credenziali Digitali sono revocate quando l'Istanza del Wallet contenitore è revocata
CI_155f	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Credenziale Digitale - Ordine Giudiziario/ Supervisorio	La Credenziale Digitale è revocata immediatamente quando attività illegali sono segnalate da Organi Giudiziari o Supervisori
CI_155g	Modello di Dati e ciclo di vita, Interoperabilità	Revoca PID - Violazione Gestore di Identità Digitale	Il PID è revocato quando si verifica rilevamento di violazione nel Gestore di Identità Digitale utilizzato per l'autenticazione Utente durante l'emissione PID
CI_156	Modello di Dati e ciclo di vita, Interoperabilità	Revoca (Q)EAA Opzionale Seguendo Revoca PID	Il Fornitore (Q)EAA valuta con successo lo stato di revoca PID e revoca le credenziali (Q)EAA associate
CI_157	Modello di Dati e ciclo di vita, Interoperabilità	Cambio Stato (Q)EAA a Sospeso	La (Q)EAA transita con successo dallo stato Emesso o Valido allo stato Sospeso quando è sospesa dal Fornitore di Attestati Elettronici ((Q)EAA SUSP)

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_158	Modello di Dati e ciclo di vita, Interoperabilità	Durata Sospensione (Q)EAA e Recupero Stato	La (Q)EAA sospesa rimane nello stato Sospeso finché le condizioni di sospensione non sono risolte e la credenziale è ripristinata allo stato precedente Emesso o Valido, o transita a Revocato, Scaduto, o eliminato
CI_158a	Modello di Dati e ciclo di vita, Interoperabilità	Sospensione (Q)EAA - Cambiamenti Validità Attributi	La sospensione di una (Q)EAA è basata sullo stato di validità dei suoi attributi contenuti
CI_158b	Modello di Dati e ciclo di vita, Interoperabilità	Sospensione (Q)EAA - Richiesta Utente	La (Q)EAA transita con successo allo stato Sospeso su richiesta esplicita dell'Utente.
CI_159	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Ciclo di Vita Individuale Credenziali Digitali Emesse in Batch	Ogni Credenziale Digitale da un'emissione batch entra con successo nella propria macchina a stati del ciclo di vita indipendente. Tutte le transizioni di stato (Emesso → Valido → Scaduto/Sospeso/ Revocato) avvengono ancora su base per Credenziale, utilizzando i parametri individuali della Credenziale
CI_160	Modello di Dati e ciclo di vita, Interoperabilità	Generazione e Memorizzazione Credenziale Digitale	Il Fornitore di Attestati Elettronici genera con successo la Credenziale Digitale seguendo il completamento della richiesta di emissione e la memorizza nell'archiviazione locale immediatamente dopo l'emissione riuscita.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_160a	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Stato Credenziale Digitale	Il Fornitore di Attestati Elettronici aggiorna con successo lo stato della Credenziale Digitale localmente per eventi di revoca, sospensione o riattivazione innescati da ragioni di sicurezza tecnica, richieste Utente, o da entità esterne (es. Utenti e Fonti Autentiche)
CI_160b	Modello di Dati e ciclo di vita, Interoperabilità	Eliminazione Dati Credenziale Digitale	Il Fornitore di Attestati Elettronici rimuove con successo le Credenziali Digitali dall'archiviazione locale dopo aver raggiunto lo stato Scaduto o basato sulle politiche di ritenzione del Fornitore di Attestati Elettronici.
CI_161	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Diretta Stato Validità Fornitore di Attestati Elettronici	Il Fornitore di Attestati Elettronici gestisce direttamente lo stato di validità delle Credenziali Digitali emesse e gestisce correttamente l'innesco del processo di revoca o sospensione della Credenziale Digitale da parte di altri attori.
CI_161a	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Avviata dall'Utente tramite Servizio Web del Fornitore di Attestati Elettronici	L'Utente avvia con successo la revoca/sospensione della Credenziale Digitale tramite il servizio web del Fornitore di Attestati Elettronici, con verifica dell'autenticazione e processamento della richiesta di revoca.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_161b	Modello di Dati e ciclo di vita, Interoperabilità	Revoca Innescata da Fonte Autentica	La Fonte Autentica innesca con successo il processo di revoca/sospensione della Credenziale Digitale quando gli attributi della Credenziale sono aggiornati o cambiano stato di validità.
CI_162	Modello di Dati e ciclo di vita, Sicurezza	Accesso Utente al Portale Web del Fornitore di Attestati Elettronici con Autenticazione LoA	Il Fornitore di Attestati Elettronici assicura che gli utenti possano accedere con successo all'area sicura del suo portale web. Questo accesso richiede autenticazione con un Livello di Garanzia almeno pari al Livello di Garanzia utilizzato durante l'emissione iniziale della credenziale.
CI_162a	Modello di Dati e ciclo di vita, Interoperabilità	Accesso Utente alla Vista Database Credenziali Digitali	Il Fornitore di Attestati Elettronici fornisce con successo agli Utenti una vista completa di tutte le loro Credenziali Digitali contenute nel database del Fornitore di Attestati Elettronici tramite il portale web
CI_162b	Modello di Dati e ciclo di vita, Interoperabilità	Verifica Autenticità Dati Utente tramite Portale Web	Il Fornitor di Credenziali abilita con successo gli Utenti a verificare l'autenticità dei dati delle loro Credenziali Digitali tramite il portale web
CI_162c	Modello di Dati e ciclo di vita, Interoperabilità	Gestione Stato Validità Utente tramite Portale Web	Il Fornitor di Credenziali consente con successo agli Utenti di visualizzare e aggiornare lo stato di validità delle loro Credenziali Digitali tramite il portale web (revocare le loro Credenziali Digitali e, se supportato dal Fornitore, sospenderle).

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_163	Modello di Dati e ciclo di vita, Interoperabilità	Revoca PID e Aggiornamento Stato Istanza del Wallet su Nuova Emissione PID	Quando l'Utente attiva un'altra Istanza di Wallet dallo stesso Fornitore di Wallet utilizzando la stessa Soluzione Wallet e ottiene un nuovo PID, il PID precedente è revocato con successo e l'Istanza di Wallet precedente transita con successo allo stato operativo, assicurando un singolo PID attivo per Utente per Fornitore di Wallet.
CI_164	Modello di Dati e ciclo di vita, Interoperabilità	Revoca PID Seguendo Morte Utente e Cambio Stato ANPR	La morte dell'Utente innesca con successo il cambio dello stato di validità degli attributi di identificazione dell'Utente nel registro pubblico (ANPR), che produce automaticamente la revoca PID.
CI_165	Modello di Dati e ciclo di vita, Interoperabilità	Requisito Notifica Revoca PID	Il Fornitore di Credenziali invia con successo notifica dell'evento di revoca PID all'Utente entro 24 ore dal verificarsi della revoca
CI_166	Modello di Dati e ciclo di vita, Interoperabilità	Notifica Revoca Credenziale Non-PID Raccomandata	Il Fornitore di Attestati Elettronici invia notifica dell'evento di revoca all'Utente per qualsiasi Credenziale Digitale diversa dal PID
CI_167	Modello di Dati e ciclo di vita, Interoperabilità	Consegna Notifica Revoca Multi-Canale	Il Fornitore di Attestati Elettronici consegna con successo le notifiche di revoca attraverso canali di comunicazione verificati e sicuri (email, telefono, o altri canali autenticati), includendo informazioni complete sullo stato di revoca della Credenziale e la ragione.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_168	Modello di Dati e ciclo di vita, Interoperabilità	Aggiornamento Stato Credenziale Digitale su Revoca	Il Fornitore di Attestati Elettronici aggiorna con successo lo stato della Credenziale Digitale immediatamente quando avviene la revoca
CI_169	Modello di Dati e ciclo di vita, Interoperabilità	Monitoraggio dello stato della Wallet Instance per l'aggiornamento dello stato degli Attestati Elettronici	Il Fornitore di Attestati Elettronici predispone un meccanismo di monitoraggio degli stati correnti di tutte le Wallet Key Attestation associate alle Wallet Instance a cui sono stati rilasciati gli Attestati Elettronici.
CI_170	Modello di Dati e ciclo di vita, Interoperabilità	Aggiornamento Stato Credenziale Seguendo Notifica Cambio Dati	Il Fornitore di Attestati Elettronici aggiorna con successo lo Stato della Credenziale secondo la modalità definita del meccanismo di validità alla ricezione di notifica dalla Fonte Autentica
CI_170a	Modello di Dati e ciclo di vita, Interoperabilità	Notifica Utente di Aggiornamento Credenziale	Il Fornitore di Attestati Elettronici notifica con successo l'Utente dei cambiamenti della credenziale attraverso canale di comunicazione out-of-band registrato
CI_171	Modello di Dati e ciclo di vita, Interoperabilità	Notifica Utente su Stato Credenziale INVALIDA	Il Fornitore di Attestati Elettronici informa con successo l'Utente quando lo Stato della Credenziale cambia a INVALIDA

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_172	Modello di Dati e ciclo di vita, Interoperabilità	Processamento Aggiornamento Stato Batch come Cambiamenti Individuali	Il Fornitore di Attestati Elettronici gestisce con successo la richiesta di aggiornamento stato batch singola (che riferisce batch notification_id) da entità autorizzate (es. Istanza di Wallet tramite Endpoint Notifica con event=credential_deleted, o Fornitore di Wallet tramite PDND) come N cambiamenti individuali separati, con lo stato di ogni Credenziale aggiornato indipendentemente (per esempio, cambiando il suo bit status-list a INVALIDO o SOSPESO).
CI_173	Modello di Dati e ciclo di vita, Interoperabilità	Revoca batch su Richiesta Aggiornamento credenziale	Il Fornitore di Attestati Elettronici processa con successo la richiesta aggiornamento batch come richiesta revoca tutto, contrassegnando ogni Credenziale nel batch come revocata ed emettendo singola notifica coprendo l'intero batch.
CI_174	Modello di Dati e ciclo di vita, Interoperabilità	Eliminazione Credenziale Batch	L'eliminazione guidata dall'utente rimuove con successo l'intero batch poiché la Wallet UI presenta un batch come una Credenziale, con richiesta eliminazione utilizzando il notification_id del batch applicato a tutte le Credenziali in quel batch, poiché non è possibile eliminare o revocare solo una Credenziale.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_175	Modello di Dati e ciclo di vita, Interoperabilità	Supporto Lista Stato OAuth per Credenziali Digitali Longeve	La Lista Stato OAuth (TOKEN-STATUS-LIST) è supportata con successo per verifica dello stato validità delle Credenziali Digitali longeve in scenari sia remoti che di prossimità.
CI_176	Modello di Dati e ciclo di vita, Interoperabilità	Allocazione Indice Credenziale Digitale e Mappatura Stato	Ogni Credenziale Digitale viene allocata con successo con un indice durante l'emissione, che rappresenta la sua posizione all'interno dell'array di bit, con il valore del bit o dei bit a questo indice che corrisponde allo stato della Credenziale Digitale.
CI_177	Modello di Dati e ciclo di vita, Interoperabilità	Formato Crittografico Status List Token	La Status List è fornita con successo all'interno del Status List Token firmato crittograficamente in formato JWT.
CI_178	Modello di Dati e ciclo di vita, Interoperabilità	Configurazione dei Bit della Status List per le Credenziali Digitali	Il Fornitore di Attestati Elettronici definisce il numero di bit, k (che può essere 1, 2, 4 o 8), che rappresenta la quantità di bit usata per descrivere lo stato di ogni Credenziale Digitale all'interno della Status List. Il Fornitore di Attestati Elettronici configura il numero di bit e, di conseguenza, ogni credenziale può avere 2^k stati possibili.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_179	Modello di Dati e ciclo di vita, Interoperabilità	Creazione di un Array di Byte per la Status List e Assegnazione della Posizione della Credenziale.	Il Fornitore di Credenziali crea con successo array byte di dimensione = (quantità di Credenziali Digitali) * k / 8 o maggiore, con ogni byte corrispondente a 8/k stati a seconda del valore k (8 se k=1, 4 se k=2, 2 se k=4, o 1 se k=8), e assegna ogni Credenziale Digitale emessa a una posizione nell'array.
CI_180	Modello di Dati e ciclo di vita, Interoperabilità	Impostazione dei valori di stato delle credenziali digitali nella Status List in un array di byte	Il Fornitore di Credenziali imposta con successo valori stato per tutte le Credenziali Digitali emesse all'interno dell'array byte, con ogni stato Credenziale Digitale identificato utilizzando un indice che mappa a bit specifici all'interno dell'array byte, conteggio indice da 0 a (quantità di Credenziale Digitale) - 1, bit contati dal bit meno significativo ("0") al bit più significativo ("7"), e tutti i bit dell'array byte a un particolare indice impostati a un valore stato.
CI_181	Modello di Dati e ciclo di vita, Interoperabilità	Livello di Compressione Raccomandato per la Status List delle Credenziali Digitali	Il Fornitore di Attestati Elettronici comprime con successo l'array byte utilizzando DEFLATE [**RFC 1951**] con il formato dati ZLIB [**RFC 1950**]
CI_181a	Modello di Dati e ciclo di vita, Interoperabilità	Livello di Compressione Raccomandato per la Status List delle Credenziali Digitali	Le implementazioni utilizzano con successo il livello di compressione più alto disponibile per la compressione dell'array di byte.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_182	Modello di Dati e ciclo di vita, Interoperabilità	Disponibilità Endpoint Status List	Il Fornitore di Attestati Elettronici rende con successo disponibile alle Relying Party e Istanze di Wallet un endpoint per richiedere Liste Stato.
CI_183	Modello di Dati e ciclo di vita, Interoperabilità	Status List Definizione dei Valori di Stato per le Credenziali Digitali	Il Fornitore di Attestati Elettronici utilizza con successo i valori per gli Stati possibili definiti nella sezione *Token Status Lists* (pagina 435).
CI_184	Modello di Dati e ciclo di vita, Interoperabilità	Status List Definizione degli stati opzionali di stato delle credenziali digitali	Il Fornitore di Attestati Elettronici aggiunge con successo altri stati oltre a quelli descritti sopra quando sceglie il numero di bit per trasmettere gli stati delle Credenziali Digitali emesse, con attenta considerazione per la divulgazione di informazioni alle Relying Party quando aggiunge molti stati diversi per il ciclo di vita della Credenziale Digitale.
CI_185	Modello di Dati e ciclo di vita, Interoperabilità	Status List Parametri del Token all'Endpoint	Il Token Status List è disponibile con successo presso lo Status List Endpoint e contiene i parametri nella *tabella* (pagina 437) corrispondente.
CI_186	Modello di Dati e ciclo di vita, Interoperabilità	Impostazione Consigliata di Scadenza Breve del Token Status List	Il Fornitore di Attestati Elettronici imposta con successo il claim exp in modo che il Token Status List sia di breve durata, tipicamente con claim exp che non supera il claim iat di più di 24 ore.
CI_187	Modello di Dati e ciclo di vita, Interoperabilità	Struttura Status List Codificata JSON	La struttura della Status List codificata JSON è conforme alla *Tabella* (pagina 438) corrispondente.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_188	Modello di Dati e ciclo di vita, Interoperabilità	Status List Archiviazione Locale delle Credenziali Digitali	I Fornitori di Attestati Elettronici memorizzano con successo la Credenziale Digitale generata localmente con set minimo di dati richiesto per gestire il suo ciclo di vita, includendo lo stato di validità di quella Credenziale Digitale.
CI_189	Modello di Dati e ciclo di vita, Interoperabilità	Inclusione nella Status List dello Stato delle Credenziali Digitali	I Fornitori di Attestati Elettronici includono con successo il claim *status_list* all'interno del valore Oggetto JSON del claim status della Credenziale Digitale una volta generata.
CI_190	Modello di Dati e ciclo di vita, Interoperabilità	Parametri dell'Oggetto JSON della Status List Claim	Il valore del claim *status_list* è con successo un Oggetto JSON con i *parametri corrispondenti* (pagina 439).
CI_191	Modello di Dati e ciclo di vita, Interoperabilità	Risposta Riuscita dello Status List Endpoint	Lo Status List Endpoint risponde con successo con il Token Status List utilizzando il codice stato HTTP nell'intervallo 2xx, con Fornitore Stato che utilizza content-type `application/statuslist+jwt` per il Token Status List in formato JWT nella risposta riuscita.
CI_192	Modello di Dati e ciclo di vita, Interoperabilità	Content-Encoding Gzip Risposta Status List HTTP	La risposta HTTP utilizza con successo il Content-Encoding gzip come definito nel [**RFC 9110**].
CI_193	Modello di Dati e ciclo di vita, Interoperabilità	Autorizzazione e Registrazione del Fornitore di Attestati Elettronici	Il Fornitore di Attestati Elettronici registra con successo le proprie Credenziali Digitali nel catalogo.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_194	Modello di Dati e ciclo di vita, Interoperabilità	Informazioni Richieste Catalogo Credenziale Digitale	Il Fornitore di Attestati Elettronici fornisce le sue credenziali nel catalogo, insieme alle informazioni nella *tabella* (pagina 206) corrispondente.
CI_195	Modello di Dati e ciclo di vita, Interoperabilità	Informazioni Elemento Array Credenziali	Ogni elemento dell'array delle credenziali contiene correttamente tutte le informazioni di primo livello definite nella *Tabella* (pagina 207) corrispondente.
CI_196	Modello di Dati e ciclo di vita, Interoperabilità	Attributi utente dell'IT-Wallet ID	L'IT-Wallet ID è emesso con gli attributi utente definiti in *Modello di Dati dell'IT-Wallet ID* (pagina 416), distinti dall'insieme di attributi del PID.
CI_197	Emissione, Autenticazione	Autenticazione primaria per IT-Wallet ID	L'EAA Provider di IT-Wallet ID autentica l'Utente con Cie ID LoA High (CIE L3) come metodo primario.
CI_198	Emissione, Autenticazione	Autenticazione eID Substantial con Verifica MRTD	Quando non si usa il PIN CieID, l'EAA Provider di IT-Wallet ID supporta il flusso di Autenticazione eID Substantial con Verifica MRTD in *Autenticazione eID Substantial con Verifica MRTD per Emissione IT-Wallet ID* (pagina 482). Gli attori di quel flusso conservano i log secondo le General Log Retention Policies.
CI_199	Emissione, Autenticazione	Authorization Details L2+	Per l'emissione di IT-Wallet ID con Verifica MRTD, l'array `authorization_details` della PAR contiene un oggetto con `type` `it_l2+document_proof` e i claim richiesti. Se quell'oggetto è assente, l'autenticazione torna a CieID LoA High.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_200	Emissione, Autenticazione, Sicurezza	Correlazione identità IdP–MRTD	L'EAA Provider lega l'identità verificata dall'IdP (Cie ID/SPID Substantial) con l'identità nell'MRTD confrontando almeno il codice fiscale o un identificativo univoco equivalente presente in entrambe le fonti. Una discordanza interrompe l'emissione. L'autenticazione primaria LoA High salta l'MRTD PoP.
CI_201	Presentazione, Sicurezza	Identity matching con identificativo nazionale univoco	Quando è presentato `personal_administrative_number` o `tax_id_code`, la Relying Party lo usa come chiave primaria di identity matching prima di qualsiasi riconciliazione con un record Utente memorizzato, come specificato in *Identity Matching e Identity Reconciliation* (pagina 567).
CI_202	Presentazione, Sicurezza	Attribute-Based Binding senza identificativo univoco	Quando un PID è presentato senza `personal_administrative_number`, l'Attribute-Based Binding non si limita a una coppia di nomi. La Relying Party confronta almeno `family_name`, `given_name`, `birth_date` (o `birthdate`) e `place_of_birth` dopo normalizzazione, e NON DEVE completare il matching su un sottoinsieme che consentirebbe uno scambio di persona, come specificato in *Identity Matching e Identity Reconciliation* (pagina 567). Binding di sessione, attested dall'Issuer o crittografico POSSONO integrare il confronto.

continues on next page

Tabella 20.33 – continua dalla pagina precedente

ID	Scopo	Descrizione	Risultato Atteso
CI_203	Emissione, Privacy	Applicabilità dell'emissione batch	Se i metadata del Credential Issuer contengono `batch_credential_issuance`, l'emissione batch si applica solo ai tipi di Credenziale per cui quei metadata sono pubblicati. L'Issuer documenta se PID e IT-Wallet ID sono in perimetro. Un tipo senza quei metadata è emesso come Credenziale singola.
CI_204	Emissione, Interoperabilità	WIA in emissione	In emissione il Credential Issuer valida una Wallet Instance Attestation (WIA), non una «Wallet Attestation» legacy. La WUA, ove usata, è un manufatto distinto.
CI_205	Emissione, Autenticazione	Mappatura della nomenclatura LoA	Nel flusso L2+, LoA3 (ISO/IEC 29115) corrisponde a eIDAS Substantial e ai mezzi nazionali L2 (SPID L2 / Cie ID Substantial). LoA High corrisponde a eIDAS High e CIE L3. Il test fallisce se un'autenticazione Substantial è registrata come High.

Matrice di Test per i Verificatori di Credenziali

Questa sezione fornisce l'insieme dei casi di test progettati per implementatori tecnici e team di sviluppo responsabili della creazione e del deployment di Verificatori di Credenziali / Relying Party.

Matrice di Test per il Verificatore di Credenziali in Remoto

Questa sezione fornisce l'insieme dei casi di test progettati per implementatori tecnici e team di sviluppo responsabili della creazione e distribuzione di soluzioni Credential Verifier per flussi remoti. È inoltre destinata agli enti di valutazione che ispezionano e validano le implementazioni delle soluzioni Credential Verifier per flussi remoti.

Nota

I riferimenti ai piani di test ufficiali OpenID4VP aggiorneranno questa sezione nelle prossime versioni.

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-01	Flusso stesso dispositivo	Verifica URL di redirect HTTP (302).	Il Relying Party emette un URL corretto utilizzando la base url fornita nei suoi metadati.
RPR-02	Flusso tra dispositivi	Verifica la generazione del QR Code per l'istanza Wallet.	Il Relying Party emette correttamente il QR Code.
RPR-03	Flusso tra dispositivi	Verifica che il QR Code contenga i parametri URL corretti.	Il Relying Party emette il QR-Code contenente un URL che utilizza la base url fornita nei suoi metadati.
RPR-04	Flusso tra dispositivi	Test della scansione del QR Code in condizioni di scarsa illuminazione.	Il QR Code viene scansionato con successo.
RPR-05	Flusso tra dispositivi	Verifica il livello di correzione degli errori del QR Code.	Il QR Code rimane leggibile anche se danneggiato.
RPR-06	Flusso tra dispositivi	Test della scansione del QR Code con dispositivi diversi.	Il QR Code viene scansionato con successo.
RPR-07	Metodo request_uri	Test di `request_uri_method` come `post`.	Il Relying Party accetta i metadati dell'istanza Wallet via POST e risponde con un oggetto Request aggiornato.
RPR-08	Metodo request_uri	Test di `request_uri_method` come `get`.	Il Relying Party emette l'oggetto Request tramite risposta HTTP GET.
RPR-09	Metodo request_uri	Test dell'assenza di `request_uri_method`.	Il Relying Party accetta e predefinisce il metodo GET.
RPR-10	Metadati	Verifica che i parametri corrispondano ai metadati OpenID Credential Verifier.	Solo i parametri consentiti saranno considerati.
RPR-11	Consenso Utente	Test dell'idoneità di un Credential Verifier nel richiedere attributi utente.	L'utente può modificare la selezione dei dati sugli attributi opzionali.
RPR-12	Risposta di Autorizzazione	Test dell'invio della Presentation Response.	Il Relying Party riceve e valida la risposta con i valori `state` e `nonce`.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-13	Risposta di Autorizzazione	Verifica la cifratura della risposta.	Il Relying Party valuta la risposta cifrata usando la propria chiave pubblica (una delle sue chiavi).
RPR-14	Gestione Errori	Test della gestione di un oggetto Request non valido.	Viene inviata una risposta di errore.
RPR-15	Gestione Errori	Verifica la registrazione degli errori.	Gli errori vengono registrati correttamente.
RPR-16	Gestione Errori	Test del recupero da errore nella richiesta di autorizzazione.	Il Relying Party invita l'utente a riprovare o a scansionare un nuovo QR code.
RPR-17	Gestione Errori	Test di un cookie HTTP falso.	Il Relying Party verifica la coerenza della sessione utente accoppiando il cookie http di sessione con state e nonce forniti.
RPR-19	Redirect URI	Test del reindirizzamento all'endpoint del Relying Party.	L'utente viene reindirizzato correttamente, l'endpoint funziona.
RPR-20	Redirect URI	Verifica la gestione di un *redirect_uri* non valido.	Viene restituita una risposta di errore.
RPR-23	Presentazione Credenziali	Verifica la conformità del formato della risposta.	Il Relying Party supporta tutti i formati di credenziali inclusi nel parametro `vp_formats_supported` dei suoi metadati.
RPR-24	Risposta di Autorizzazione	Test della gestione dei timeout di risposta.	I tentativi devono avere successo a meno che non venga acquisita la risposta.
RPR-25	Gestione Errori	Verifica la gestione di claim malformati nel payload della presentazione.	Viene inviata una risposta di errore di autorizzazione.
RPR-26	Gestione Errori	Verifica la gestione di claim malformati nelle credenziali presentate.	Viene inviata una risposta di errore di autorizzazione.
RPR-27	Gestione Errori	Test della gestione di richieste scadute.	Il titolare viene informato della scadenza.
RPR-28	Risposta del Relying Party	Verifica l'inclusione del codice di risposta.	Il codice di risposta è crittograficamente casuale.
RPR-29	Risposta del Relying Party	Test della gestione di codici di risposta non validi.	Viene restituita una risposta di errore.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-30	Endpoint di Stato	Verifica la gestione di accessi non autorizzati.	L'accesso non autorizzato viene negato.
RPR-31	Endpoint di Stato	Test della gestione di ID di sessione non validi.	Viene restituita una risposta di errore.
RPR-32	Redirect URI	Verifica la gestione di sessioni scadute.	Viene restituita una risposta di errore.
RPR-33	Redirect URI	Test della gestione di errori del server.	Viene restituita una risposta di errore.
RPR-34	Flusso stesso e tra dispositivi	Verifica la gestione di condizioni di rete lente.	Il Relying Party fornisce la risposta http entro il limite massimo di 2 secondi.
RPR-36	Risposta di Presentazione	Verifica la gestione di payload di risposta di grandi dimensioni.	La risposta viene valutata entro soglie di sicurezza appropriate.
RPR-37	Risposta di Presentazione	Test della gestione di errori di cifratura della risposta.	Viene restituita una risposta di errore.
RPR-38	Gestione Errori	Verifica la gestione di firme non valide.	Viene inviata una risposta di errore di autorizzazione.
RPR-39	Gestione Errori	Test della gestione di valori nonce non validi.	Viene restituita una risposta di errore.
RPR-40	Risposta del Relying Party	Verifica la gestione di risposte malformate.	Viene restituita una risposta di errore.
RPR-41	Risposta del Relying Party	Test della gestione di parametri di risposta mancanti.	Viene restituita una risposta di errore.
RPR-42	Endpoint di Stato	Verifica la gestione dei timeout di sessione.	Viene restituita una risposta di errore.
RPR-43	Endpoint di Stato	Test della gestione di codici di stato non validi.	Viene restituita una risposta di errore.
RPR-44	Redirect URI	Verifica la gestione di sessioni utente non valide.	Viene restituita una risposta di errore.
RPR-45	Redirect URI	Test della gestione di servizi non disponibili.	Viene restituita una risposta di errore.
RPR-46	Flusso stesso dispositivo	Verifica la gestione delle cancellazioni da parte dell'utente.	L'utente può annullare il processo.
RPR-47	Flusso tra dispositivi	Test della scansione del QR Code con app diverse.	Il QR Code viene scansionato con successo.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-48	Flusso tra dispositivi	Verifica la scansione del QR Code con illuminazione diversa.	Il QR Code viene scansionato con successo.
RPR-49	Metodo request_uri	Test della gestione di content type non supportati.	Viene restituita una risposta di errore.
RPR-50	Consenso Utente	Verifica la notifica all'utente delle modifiche al consenso.	L'utente viene informato delle modifiche al consenso.
RPR-51	Consenso Utente	Test del consenso dell'utente per dati sensibili.	L'utente può acconsentire ai dati sensibili.
RPR-52	Risposta di Autorizzazione	Verifica la gestione di errori di decifratura della risposta.	Viene restituita una risposta di errore.
RPR-53	Risposta di Autorizzazione	Test della verifica dell'integrità della risposta.	L'integrità della risposta è verificata.
RPR-54	Risposta del Relying Party	Verifica la gestione di errori di validazione della risposta.	Viene restituita una risposta di errore.
RPR-55	Risposta del Relying Party	Test della gestione di errori di elaborazione della risposta.	Viene restituita una risposta di errore.
RPR-56	Endpoint Risorsa Protetta	Verifica la gestione di accessi di sessione non autorizzati.	L'accesso non autorizzato viene negato.
RPR-57	Redirect URI	Verifica la gestione di parametri di redirect non validi.	Viene restituita una risposta di errore.
RPR-58	Redirect URI	Test della gestione di errori di redirect.	Viene restituita una risposta di errore.
RPR-59	Flusso stesso dispositivo	Verifica la gestione di interruzioni da parte dell'utente.	L'utente può riprendere o annullare il processo.
RPR-60	Metodo request_uri	Test della gestione di metodi HTTP non validi.	Viene restituita una risposta di errore.
RPR-61	Consenso Utente	Verifica la notifica all'utente della revoca del consenso.	L'utente viene informato della revoca del consenso.
RPR-62	Consenso Utente	Test del consenso dell'utente per dati opzionali.	L'utente può acconsentire ai dati opzionali.
RPR-63	Risposta di Autorizzazione	Verifica la gestione di errori di firma della risposta.	Viene restituita una risposta di errore.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-64	Risposta di Autorizzazione	Test della gestione di errori di formato della risposta.	Viene restituita una risposta di errore.
RPR-65	Gestione Errori	Verifica la gestione di firme JWT non valide.	Viene inviata una risposta di errore di autorizzazione.
RPR-66	Gestione Errori	Test della gestione di claim JWT non validi.	Viene restituita una risposta di errore.
RPR-67	Risposta del Relying Party	Verifica la gestione di errori di parsing della risposta.	Viene restituita una risposta di errore.
RPR-68	Risposta del Relying Party	Test della gestione di errori di timeout della risposta.	Viene restituita una risposta di errore.
RPR-69	Endpoint di Stato	Verifica la gestione della scadenza della sessione.	Viene restituita una risposta di errore.
RPR-70	Endpoint di Stato	Test della gestione di errori di rinnovo della sessione.	Viene restituita una risposta di errore.
RPR-71	Redirect URI	Verifica la gestione di errori di loop di redirect.	Viene restituita una risposta di errore.
RPR-72	Redirect URI	Test della gestione di errori di sicurezza nel redirect.	Viene restituita una risposta di errore.
RPR-73	Flusso stesso dispositivo	Verifica la gestione dei timeout utente.	L'utente viene informato del timeout.
RPR-74	Flusso tra dispositivi	Test della scansione del QR Code con dispositivi diversi.	Il QR Code viene scansionato con successo.
RPR-75	Flusso tra dispositivi	Verifica la scansione del QR Code con app diverse.	Il QR Code viene scansionato con successo.
RPR-76	Metodo request_uri	Test della gestione di metodi HTTP non supportati.	Viene restituita una risposta di errore.
RPR-77	Generazione QR Code	Verifica che il livello di correzione degli errori del QR Code sia Q (Quartile - fino al 25%).	Il QR Code utilizza il livello di correzione Q richiesto.
RPR-78	Richiesta Wallet Instance Attestation (WIA)	Test che la richiesta di Wallet Instance Attestation (WIA) utilizzi la query DCQL standard.	La richiesta di Wallet Instance Attestation (WIA) utilizza correttamente la query DCQL standard.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-79	Richiesta Wallet Instance Attestation (WIA)	Verifica che il parametro `claims` non sia incluso nella query DCQL per l'Attestato dell'Istanza del Wallet (WIA).	Il parametro `claims` non è incluso nella query DCQL per l'Attestato dell'Istanza del Wallet (WIA).
RPR-80	Richiesta Wallet Instance Attestation (WIA)	Test che il parametro `vct_values` sia richiesto nella query DCQL per l'Attestato dell'Istanza del Wallet (WIA).	Il parametro `vct_values` è correttamente richiesto nella query DCQL.
RPR-81	Wallet Nonce	Test che il Relying Party controlli `wallet_nonce` quando presente.	Il Relying Party controlla correttamente `wallet_nonce` quando presente.
RPR-82	Tipi di Risposta	Verifica che `response_types_supported` sia impostato a `vp_token` quando presente.	`response_types_supported` è correttamente impostato a `vp_token`.
RPR-83	Redirect URI	Test che il Relying Party fornisca correttamente il parametro `redirect_uri` all'istanza Wallet.	Il Relying Party fornisce e gestisce correttamente `redirect_uri`.
RPR-84	Supporto Flussi	Test che il Relying Party supporti i flussi remoti richiesti.	Il Relying Party supporta sia il flusso stesso dispositivo che tra dispositivi.
RPR-85	Sicurezza Endpoint	Test che `request_uri` sia attestato da una terza parte fidata.	Il parametro `request_uri` è correttamente attestato da una terza parte fidata.
RPR-86	Protezione della Privacy	Test che il Relying Party validi correttamente i metadati dell'istanza Wallet senza informazioni utente.	Il Relying Party valuta correttamente le capacità tecniche dell'istanza Wallet.
RPR-87	Metodo POST request_uri	Test che il Relying Party supporti la ricezione dei metadati dell'istanza Wallet via POST all'endpoint `request_uri` con content type `application/x-www-form-urlencoded`.	Il Relying Party accetta e processa correttamente i metadati dell'istanza Wallet inviati via POST con il content type richiesto.
RPR-88	Validazione Algoritmo	Test che l'algoritmo JWT sia supportato e non sia `none` o MAC.	L'algoritmo JWT è tra quelli supportati e non è `none` o un identificatore MAC.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-89	Validazione Media Type	Test che JWT typ sia impostato a `oauth-authz-req+jwt`.	Il parametro typ di JWT è correttamente impostato a `oauth-authz-req+jwt`.
RPR-90	Validazione Response Mode	Test che `response_mode` sia impostato a `direct_post.jwt` sia nel Flusso Same Device sia nel Flusso Cross Device.	Il parametro `response_mode` è correttamente impostato a `direct_post.jwt` per ogni transazione di presentazione remota.
RPR-91	Validazione Response Type	Test che `response_type` sia impostato a `vp_token`.	Il parametro `response_type` è correttamente impostato a `vp_token`.
RPR-92	Uso di Response URI	Test che il Relying Party fornisca correttamente il parametro `response_uri` all'istanza Wallet.	Il Relying Party invia la risposta di autorizzazione al corretto endpoint `response_uri`.
RPR-93	Entropia del Nonce	Test che il nonce abbia sufficiente entropia (32+ cifre).	Il parametro nonce ha sufficiente entropia con almeno 32 cifre.
RPR-94	Scadenza JWT	Test che il parametro `exp` di JWT sia impostato correttamente.	Il parametro `exp` di JWT è correttamente impostato e non scaduto.
RPR-95	Sicurezza Response URI	Test che `response_uri` sia attestato da una terza parte fidata.	Il parametro `response_uri` è correttamente attestato da una terza parte fidata.
RPR-96	Gestione Client Metadata	Test che il Relying Party gestisca correttamente il `client_metadata` dell'istanza Wallet.	Il `client_metadata` è correttamente allineato con i metadati della Trust Chain.
RPR-97	Richiesta Wallet Instance Attestation (WIA)	Test che il Relying Party richieda l'Attestato dell'Istanza del Wallet (WIA) tramite DCQL.	Il Relying Party richiede correttamente l'Attestato dell'Istanza del Wallet (WIA) usando la query DCQL.
RPR-98	Formato Risposta di Errore	Test che la risposta di errore utilizzi il content type `application/json`.	La risposta di errore utilizza correttamente il content type `application/json`.
RPR-99	Parametri Risposta di Errore	Test che la risposta di errore includa i parametri richiesti.	La risposta di errore include i parametri error ed `error_description`.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-100	Presentazione Wallet Instance Attestation (WIA)	Test che il Relying Party richieda correttamente l'Attestato dell'Istanza del Wallet (WIA) all'istanza Wallet.	Il Relying Party valuta correttamente l'Attestato dell'Istanza del Wallet (WIA) quando richiesta.
RPR-101	Array Presentazione	Test che `vp_token` contenga le presentazioni firmate richieste.	`vp_token` contiene le presentazioni firmate richieste come previsto.
RPR-102	Inclusione KB-JWT	Test che il Titolare includa KB-JWT in SD-JWT.	Il Titolare include correttamente KB-JWT nella presentazione SD-JWT.
RPR-103	Validazione KB-JWT	Test che il Relying Party validi la firma KB-JWT.	Il Relying Party valida correttamente la firma KB-JWT usando la chiave pubblica.
RPR-104	Header KB-JWT	Test che KB-JWT contenga i parametri header richiesti.	KB-JWT contiene i parametri header `typ` e `alg` richiesti.
RPR-105	Payload KB-JWT	Test che KB-JWT contenga i parametri payload richiesti.	KB-JWT contiene i parametri `iat`, `aud`, `nonce` e `sd_hash` richiesti.
RPR-106	Audience KB-JWT	Test che `aud` di KB-JWT corrisponda all'identificativo del Relying Party.	Il parametro `aud` di KB-JWT corrisponde all'identificativo univoco del Relying Party.
RPR-107	Nonce KB-JWT	Test che `nonce` di KB-JWT corrisponda al `nonce` dell'oggetto request.	Il parametro `nonce` di KB-JWT corrisponde al `nonce` dell'oggetto request.
RPR-108	Risposta di Errore di Autorizzazione	Test che il Relying Party gestisca correttamente la risposta di errore di autorizzazione dell'istanza Wallet in caso di fallimento della validazione.	Il Relying Party valuta correttamente la risposta di errore di autorizzazione quando la validazione fallisce.
RPR-109	Codifica Risposta di Errore	Test che la risposta di errore di autorizzazione sia codificata correttamente.	La risposta di errore di autorizzazione è codificata in formato `application/x-www-form-urlencoded`.
RPR-110	Elaborazione Risposta	Test che la Response URI restituisca HTTP 200 in caso di elaborazione corretta.	La Response URI restituisce HTTP 200 con content type `application/json`.

continues on next page

Tabella 20.34 – continua dalla pagina precedente

ID Caso di Test	Scopo	Descrizione	Risultato Atteso
RPR-111	Coerenza Codici di Errore	Test che i codici di errore siano coerenti tra i diversi endpoint.	I codici di errore sono coerenti su tutti gli endpoint del Relying Party.
RPR-112	Inclusione Codice di Risposta	Test che il Relying Party includa il codice di risposta in `redirect_uri`.	Il Relying Party include un nuovo codice di risposta in `redirect_uri`.
RPR-113	Sicurezza Redirect URI	Test che `redirect_uri` sia attestato da una terza parte fidata.	Il parametro `redirect_uri` è correttamente attestato da una terza parte fidata.
RPR-114	Risposta di Errore di Validazione	Test che la Response URI restituisca una risposta di errore in caso di fallimento della validazione.	La Response URI restituisce una risposta di errore quando i controlli di validazione falliscono.
RPR-115	Identity Matching	Test dell'Unique National Identifier Binding quando è presentato `personal_administrative_number` o `tax_id_code`.	La Relying Party usa l'identificativo nazionale univoco come chiave primaria di identity matching prima della riconciliazione, come specificato in *Identity Matching e Identity Reconciliation* (pagina 567) (*CI_201* (pagina 825)).
RPR-116	Identity Matching	Test dell'Attribute-Based Binding quando un PID è presentato senza `personal_administrative_number`.	La Relying Party confronta almeno `family_name`, `given_name`, `birth_date` (o `birthdate`) e `place_of_birth` dopo normalizzazione e NON DEVE completare il matching su un sottoinsieme che consentirebbe uno scambio di persona, come specificato in *Identity Matching e Identity Reconciliation* (pagina 567) (*CI_202* (pagina 825)).

Matrice di Test per il Verificatore di Credenziali in Prossimità

Questa sezione fornisce l'insieme dei casi di test progettati per implementatori tecnici e team di sviluppo responsabili della creazione e del deployment di soluzioni Credential Verifier per i flussi di prossimità. È anche destinata agli organismi di valutazione che ispezionano e validano le implementazioni di soluzioni Credential Verifier per i flussi di prossimità.

Nota

Ulteriori riferimenti sui piani di test ufficiali ISO-18013-5, se disponibili, aggiorneranno questa sezione nelle versioni future.

Test Case ID	Purpose	Description	Expected Result
PPR-001	Device Engagement	Testare l'avvio del device engagement utilizzando QR code.	Il device engagement viene avviato con successo e il QR code viene scansionato.
PPR-002	Session Establishment	Verificare l'instaurazione della sessione con le session key corrette.	La sessione viene stabilita in modo sicuro con le session key corrette.
PPR-003	Communication	Testare la trasmissione della mdoc request tramite BLE.	La mdoc request viene trasmessa in modo sicuro tramite BLE.
PPR-004	User Authentication	Validare l'autenticazione dell'utente tramite WSCA.	L'utente viene autenticato con successo utilizzando WSCA.
PPR-005	Attribute Consent	Verificare il consenso dell'utente per il rilascio degli attributi.	L'utente acconsente al rilascio degli attributi richiesti.
PPR-006	Data Retrieval	Testare il recupero delle mdoc Digital Credentials.	Le mdoc Digital Credentials vengono recuperate con successo.
PPR-007	Session Termination	Verificare la terminazione della sessione dopo lo scambio di dati.	La sessione viene terminata e le chiavi vengono distrutte.
PPR-008	Error Handling	Testare la gestione di session key non valide.	Viene visualizzato un messaggio di errore appropriato per chiavi non valide.
PPR-009	BLE Connection	Testare la stabilità della connessione BLE durante lo scambio di dati.	La connessione BLE rimane stabile durante tutto lo scambio.

continues on next page

Tabella 20.35 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
PPR-010	Document Verification	Verificare l'integrità dei documenti ricevuti.	I documenti vengono verificati e l'integrità è confermata.
PPR-011	Security	Testare la crittografia delle mdoc request e response.	Tutte le mdoc request e response vengono crittografate correttamente.
PPR-012	User Interface	Verificare l'interfaccia utente per il consenso agli attributi.	L'interfaccia utente visualizza chiaramente la richiesta di consenso agli attributi.
PPR-013	Error Handling	Testare la risposta a tipi di documento non supportati.	Il sistema restituisce un errore appropriato per tipi di documento non supportati.
PPR-014	Performance	Misurare il tempo impiegato per l'instaurazione della sessione.	La sessione viene stabilita entro limiti di tempo accettabili.
PPR-015	Compatibility	Verificare la compatibilità con diversi dispositivi mobili.	Il sistema funziona senza problemi su vari dispositivi mobili.
PPR-016	Data Integrity	Testare l'integrità dei dati durante la trasmissione.	L'integrità dei dati viene mantenuta durante la trasmissione.
PPR-017	Session Management	Testare la gestione delle sessioni sotto carico elevato.	Le sessioni vengono gestite efficacemente in condizioni di carico elevato.
PPR-018	BLE Connection	Testare la riconnessione dopo la disconnessione BLE.	Il sistema si riconnette con successo dopo la disconnessione BLE.
PPR-019	User Experience	Valutare l'esperienza utente durante il flusso di prossimità.	Gli utenti riportano un'esperienza positiva con il flusso di prossimità.
PPR-020	Security	Testare la resistenza agli attacchi replay.	Il sistema è resistente agli attacchi replay.
PPR-021	Device Engagement	Verificare che la struttura Device Engagement sia codificata in CBOR.	La struttura Device Engagement è correttamente codificata in formato CBOR.
PPR-022	Device Engagement	Testare che la chiave pubblica effimera sia del tipo consentito dalla suite di cifratura.	La chiave pubblica effimera rispetta i requisiti della suite di cifratura selezionata.
PPR-023	BLE Configuration	Verificare che solo la modalità Central Client sia supportata.	La modalità Central Client è supportata per le connessioni BLE.

continues on next page

Tabella 20.35 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
PPR-024	Capabilities	Testare che `HandoverSessionEstablishmentSupport` sia impostato su `true` quando presente.	`HandoverSessionEstablishmentSupport` è correttamente impostato su `true`.
PPR-025	Capabilities	Verificare che `ReaderAuthAllSupport` sia impostato su `true` quando presente.	`ReaderAuthAllSupport` è correttamente impostato su `true`.
PPR-026	mdoc Request	Testare che i messaggi mdoc Request siano codificati in CBOR.	I messaggi mdoc Request sono correttamente codificati in formato CBOR.
PPR-027	mdoc Request	Verificare che la richiesta mdoc sia cifrata con la chiave di sessione.	La richiesta mdoc è correttamente cifrata con la chiave di sessione.
PPR-028	mdoc Request	Testare che la richiesta mdoc sia trasmessa tramite protocollo BLE.	La richiesta mdoc è trasmessa correttamente tramite protocollo BLE.
PPR-029	mdoc Response	Verificare che i messaggi mdoc Response siano codificati in CBOR.	I messaggi mdoc Response sono correttamente codificati in formato CBOR.
PPR-030	mdoc Response	Testare che la risposta mdoc sia cifrata con la chiave di sessione.	La risposta mdoc è correttamente cifrata con la chiave di sessione.
PPR-031	Key Management	Testare che la chiave effimera privata sia mantenuta segreta.	La chiave effimera privata è adeguatamente protetta e non esposta.
PPR-032	Key Management	Testare che la chiave effimera pubblica sia usata nell'instaurazione della sessione.	La chiave effimera pubblica è utilizzata correttamente per l'instaurazione della sessione.
PPR-033	Derivazione Chiavi di Sessione	Testare che le chiavi di sessione siano derivate con il protocollo di accordo chiavi.	Le chiavi di sessione sono derivate correttamente usando il protocollo di accordo chiavi.
PPR-034	Session Establishment	Testare che il messaggio `SessionEstablishment` sia preparato correttamente.	Il messaggio `SessionEstablishment` è preparato correttamente con i componenti richiesti.
PPR-035	Session Establishment	Testare che il messaggio `SessionEstablishment` sia firmato dalla RP Instance.	Il messaggio `SessionEstablishment` è firmato correttamente dalla Relying Party Instance.

continues on next page

Tabella 20.35 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
PPR-036	Session Establishment	Testare che il messaggio `SessionEstablishment` sia cifrato con le chiavi di sessione.	Il messaggio `Session Establishment` è correttamente cifrato con le chiavi di sessione.
PPR-037	Session Establishment	Testare che `Session Establishment` includa `EReader Key.Pub` e la richiesta di attributi.	Il messaggio `Session Establishment` include `EReader Key.Pub` e la richiesta di attributi richiesti.
PPR-038	Trasmissione Messaggi	Testare che `Session Establishment` sia trasmesso su connessione BLE sicura.	Il messaggio `Session Establishment` è trasmesso su connessione BLE sicura.
PPR-039	Calcolo Chiave di Sessione	Testare che l'Istanza del Wallet calcoli correttamente la chiave di sessione.	L'Istanza del Wallet calcola correttamente la chiave di sessione.
PPR-040	Decrittazione Messaggi	Testare che l'Istanza del Wallet decritti il messaggio `SessionEstablishment`.	L'Istanza del Wallet decritta con successo il messaggio `SessionEstablishment`.
PPR-041	Verifica Firma	Testare che l'Istanza del Wallet verifichi la firma della RP Instance.	L'Istanza del Wallet verifica correttamente la firma della Relying Party Instance.
PPR-042	Elaborazione Richiesta Attributi	Testare che l'Istanza del Wallet decritti la richiesta di attributi.	L'Istanza del Wallet decritta con successo la richiesta di attributi.
PPR-043	Consenso Utente	Testare che l'Istanza del Wallet richieda il consenso all'utente.	L'Istanza del Wallet richiede correttamente il consenso al rilascio degli attributi.
PPR-044	Visualizzazione Certificato	Testare che l'Istanza del Wallet mostri il Certificato di Registrazione della RP.	L'Istanza del Wallet mostra il Certificato di Registrazione della Relying Party per trasparenza.
PPR-045	Recupero Credenziali	Testare che l'Istanza del Wallet recuperi le mdoc Digital Credentials richieste.	L'Istanza del Wallet recupera con successo le mdoc Digital Credentials richieste.
PPR-046	Preparazione `Session Data`	Testare che l'Istanza del Wallet prepari il messaggio `SessionData`.	L'Istanza del Wallet prepara correttamente il messaggio `SessionData` con le Digital Credentials.
PPR-047	Firma Dati di Autenticazione	Testare che l'Istanza del Wallet firmi i dati di autenticazione richiesti.	L'Istanza del Wallet firma correttamente i dati di autenticazione richiesti.
PPR-048	Cifratura Messaggi	Testare che `SessionData` sia cifrato con le chiavi di sessione.	Il messaggio `SessionData` è correttamente cifrato con le chiavi di sessione.

continues on next page

Tabella 20.35 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
PPR-049	Codifica CBOR	Testare che la response mdoc sia codificata in formato CBOR.	La response mdoc è correttamente codificata in formato CBOR.
PPR-050	Verifica Dati	Testare che la RP Instance decritti `SessionData`.	La Relying Party Instance decritta con successo `SessionData`.
PPR-051	Verifica Firma	Testare che la RP Instance verifichi la firma dell'Istanza del Wallet.	La Relying Party Instance verifica correttamente la firma dell'Istanza del Wallet.
PPR-052	Validazione Documenti	Testare che la RP Instance controlli la validità dell'mdoc e la firma dell'Issuer.	La Relying Party Instance valida correttamente l'mdoc e la firma dell'Issuer.
PPR-053	Disconnessione BLE	Testare che il GATT Client si disiscriva dalle caratteristiche.	Il GATT Client si disiscrive correttamente dalle caratteristiche.
PPR-054	Disconnessione BLE	Testare che il GATT Client si disconnetta dal server GATT.	Il GATT Client si disconnette correttamente dal server GATT.
PPR-055	Conformità Struttura Request	Testare che la mdoc Request sia conforme alla struttura richiesta.	La mdoc Request è conforme alla struttura richiesta e include i componenti necessari.
PPR-056	Conformità Struttura Response	Testare che la mdoc Response sia conforme alla struttura richiesta.	La mdoc Response è conforme alla struttura richiesta e include i componenti necessari.
PPR-057	Conformità Struttura Documenti	Testare che i documenti siano conformi alla struttura richiesta.	I documenti sono conformi alla struttura richiesta e includono i componenti necessari.
PPR-058	Validazione Tipo Documento	Testare che il tipo documento mDL sia impostato correttamente.	Il tipo documento mDL è impostato correttamente su `org.iso.18013.5.1.mDL`.
PPR-059	Struttura Device Signed	Testare che la struttura `deviceSigned` sia conforme.	La struttura `deviceSigned` è conforme al formato richiesto e include i componenti necessari.
PPR-060	Autenticazione Dispositivo	Testare che `deviceAuth` includa `deviceSignature`.	La struttura `deviceAuth` include `deviceSignature` richiesto per l'autenticazione.
PPR-061	Inclusione Wallet Instance Attestation (WIA)	Testare che l'Istanza del Wallet includa la Wallet Instance Attestation (WIA) quando richiesto.	L'Istanza del Wallet include la Wallet Instance Attestation (WIA) quando richiesto dalla Relying Party.

continues on next page

Tabella 20.35 – continua dalla pagina precedente

Test Case ID	Purpose	Description	Expected Result
PPR-062	Inclusione Claim AAL	Testare che l'Istanza del Wallet includa il claim `aal` nella Wallet Instance Attestation (WIA).	L'Istanza del Wallet include il claim `aal` come disclosure nella Wallet Instance Attestation (WIA).
PPR-063	Bypass Consenso Utente	Testare che l'Istanza del Wallet non richieda il consenso utente per la Wallet Instance Attestation (WIA).	L'Istanza del Wallet non richiede il consenso per gli attributi tecnici della Wallet Instance Attestation (WIA).
PPR-064	Condizioni di Terminazione Sessione	Testare che la sessione venga terminata alle condizioni specificate.	La sessione è terminata correttamente quando si verificano le condizioni specificate.
PPR-065	Avvio Terminazione Sessione	Testare che la terminazione della sessione sia avviata correttamente.	La terminazione della sessione è avviata correttamente quando non vengono inviate ulteriori richieste.
PPR-066	Distruzione Chiavi	Testare che le chiavi di sessione siano distrutte alla terminazione.	Le chiavi di sessione e il materiale effimero sono distrutti correttamente.
PPR-067	Chiusura Canale	Testare che il canale di comunicazione sia chiuso alla terminazione.	Il canale di comunicazione usato per il recupero dati è chiuso correttamente.

20.5 Specifica OpenAPI del Fornitore di Wallet PDND

Di seguito è riportata la specifica OpenAPI completa per i servizi elettronici PDND del Fornitore di Wallet:

```yaml
openapi: 3.0.1
info:
  title: IT Wallet API - Wallet Provider web services
  version: 0.2.0
  description: IT Wallet Provider e-Service exposed via PDND.
  termsOfService: "https://authentic-source.example.it/tos/"
  contact:
    name: IT-Wallet <wallet_provider>
    url: https://github.com/italia/eid-wallet-it-docs
  x-api-id: WPITW-01
  x-summary: IT Wallet Wallet Provider API.
servers:
  - url: https://test.wallet-provider.example.it/v0.2.0
    description: Wallet Provider API test server
```

```yaml
  - url: https://wallet-provider.example.it/v0.2.0
    description: Wallet Provider API production server
paths:
  /status:
    get:
      tags:
        - status
      summary: Get Wallet Provider API status.
      description: Health-check endpoint that returns the operational status of the Wallet Provider API.
      operationId: walletProviderStatus
      responses:
        "200":
          description: Service available
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            Cache-Control:
              $ref: "#/components/headers/CacheControlHeader"
            RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "429":
          description: Too Many Requests
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
            RateLimit-Limit:
              $ref: "#/components/headers/RateLimitLimitHeader"
            RateLimit-Remaining:
              $ref: "#/components/headers/RateLimitRemainingHeader"
            RateLimit-Reset:
              $ref: "#/components/headers/RateLimitResetHeader"
        "503":
          description: Service Unavailable
          content:
            application/problem+json:
              schema:
                $ref: "#/components/schemas/ProblemDetails"
          headers:
```

```
61            Retry-After:
62              $ref: "#/components/headers/RetryAfterHeader"
63
64    /wallet-instances:
65      patch:
66        tags:
67          - instance
68        summary: Notification of User's death.
69        description: >-
70          This service is used to notify the Wallet Provider of the need to revoke the Wallet Ins
    tance and delete the User's account due to the User's death.
71        operationId: notifyUserDeath
72        parameters:
73          - name: Agid-JWT-Signature
74            in: header
75            description: >-
76              JWT containing the signature of the message headers whose integrity
77              needs to be guaranteed, to comply with the INTEGRITY_REST_02
78              security pattern (see <a target="blank"
79              href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pd
    nd.html">e-Service PDND</a>). <br/><br/>
80
81              <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZ
    CI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi0
    2MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5Mz
    MiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxN
    zMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZjdiMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIs
    InNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2PTcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1Yzl
    mMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNvbnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.t
    G5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5Sm2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA">EXAMPLE
82              ON JWT.IO</a>
83            required: true
84            schema:
85              type: string
86              format: JWT
87              example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NT
    QzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJzdWIiOiI4
    MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXh
    hbXBsZS5pdCIsImlhdCI6MTczMzM5Nzg0MCwibmJmIjoxNzMzNDAxNjI4LCJleHAiOjE3MzM0MDE0NDAsImp0aSI6ImQzZj
    diMmM5LTI3NGEtNDJiNy04ZjhkLTJlOWQ4YjE3MzRiMCIsInNpZ25lZF9oZWFkZXJzIjpbeyJkaWdlc3QiOiJTSEEtMjU2P
    TcyZTE4YmRkZGYxM2M5MTFiNGRkNTYyZWUyMTk3OWE1YzlmMjM1YzNhMDFiZDE0MjZlODU3ZDhjMWEyODJmNDEifSx7ImNv
    bnRlbnQtdHlwZSI6ImFwcGxpY2F0aW9uL2pzb24ifV19.tG5-P96CCA6N1IYC-xk4GumoVkA3NFolpbBn2vQ2e9vpWQ8f5S
    m2l4-1VrXfKTx-CUVz_puiwqkBhulrNKj2fA
88          - name: Digest
89            in: header
90            description: >-
91              Digest of the message payload, to comply with the INTEGRITY_REST_02
```

```
92              security pattern. According to <a target="blank" href="https://www.rfc-editor.org/rfc/rfc3230.html#section-4.2">RFC
93              3230 §4.2</a>, the format MUST be the following: digest-algorithm=encoded
94              digest output.
95            required: true
96            schema:
97              type: string
98              example: SHA-256=72e18bdddf13c911b4dd562ee21979a5c9f235c3a01bd1426e857d8c1a282f41
99          - name: Agid-JWT-TrackingEvidence
100           in: header
101           description: >-
102             TrackingEvidence JWT for AUDIT_REST_02. Optional. It MAY be sent
103             together with POP_DPoP (`Authorization: DPoP` and the `DPoP` proof)
104             when the Provider requires tracked data from the Consumer, such as
105             caller LoA, userID, IP address, or other data agreed with the
106             Provider. The DPoP proof remains the proof of possession. When the
107             Provider mandates AUDIT_REST_02, this header MUST be present.
108             For POP_TPoP the same header is required by `TrackingEvidenceAuth`
109             together with `BearerAuth`; send a single header value
110             (see <a target="blank"
111             href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pdnd.html">e-Service PDND</a>). <br/><br/>
112             <a target="blank" href="https://jwt.io/#debugger-io?token=eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA">EXAMPLE
113             ON JWT.IO</a>
114           required: false
115           schema:
116             type: string
117             format: JWT
118             example: eyJhbGciOiJFUzI1NiIsImtpZCI6ImQ0YzNiMmExLTk4NzYtNTQzMi0xMGZlLWRjYmE5ODc2NTQzMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI4MjkxNGIzZi02MGIyLTQ1MjktYjRkNi0zZDRlNjdmMGE5MzMiLCJhdWQiOiJodHRwczovL2F1dGhlbnRpYy1zb3VyY2UuZXhhbXBsZS5pdCIsImV4cCI6MTczMzA1MjYwMCwibmJmIjoxNzMzMDM2NDUwLCJpYXQiOjE3MzMwMzY0MDAsImp0aSI6ImE0YjVjNmQ3LWU4ZjktYWJjZC1lZjEyLTM0NTY3ODkwMTIzNCIsImRub25jZSI6NjUyODQyNDIxMzY4NSwicHVycG9zZUlkIjoiYjJjM2Q0ZTUtZjZnNy1oOGk5LWowazEtbG1ubzEyMzQ1Njc4IiwidXNlcklEIjoiYThiN2M2ZDUtZTRmMy1nMmgxLWk5ajAta2xtbm9wcXJzdHV2IiwibG9hIjoic3Vic3RhbnRpYWwifQ.y42yfMeW2H9h0b0j0BODUml8yF20stY9q3BwoVU5BB90afBj852Q0QlInncdhjXhUjLS1V76cGBxkutDNvxRNA
119       security:
120         - DPoPVoucherAuth: []
121           DPoPProof: []
122         - BearerAuth: []
123           TrackingEvidenceAuth: []
```

```
124       requestBody:
125         required: true
126         content:
127           application/merge-patch+json:
128             schema:
129               $ref: "#/components/schemas/NotifyUserDeath"
130       responses:
131         "207":
132           description: Multi-Status
133           headers:
134             Agid-JWT-Signature:
135               description: JWT containing the signature of the message headers whose integrity
    needs to be guaranteed, to comply with the INTEGRITY_REST_02 security pattern (see <a target="b
    lank" href="https://italia.github.io/eid-wallet-it-docs/versione-corrente/en/e-service-pdnd.htm
    l">e-Service PDND</a>). <a target="blank" href="https://jwt.io/#debugger-io?token=ew0KICAiYWxnI
    jogIkVTMjU2IiwNCiAgImtpZCI6ICJhMWY1YzhkMi00YjM3LTRlOTEtYjBkMi03OWUzZjBjNGE4ZWYiLA0KICAidHlwIjog
    IkpXVCINCn0.ew0KICAiaXNzIjogIjEyMzRhYmNkLWVmNTYtZ2g3OC1pOWowLWtsbW5vcHFyc3R3eCIsDQogICJzdWIiOiA
    iMTIzNGFiY2QtZWY1Ni1naDc4LWk5ajAta2xtbm9wcXJzdHd4IiwNCiAgImF1ZCI6ICJodHRwczovL2ZydWl0b3JlLmV4YW
    1wbGUvZW50ZS1leGFtcGxlL3YxIiwNCiAgImlhdCI6IDE3MzMzOTc4NDAsDQogICJuYmYiOiAxNzMzNDAxNjI4LA0KICAiZ
    XhwIjogMTczMzQwMTQ0MCwNCiAgImp0aSI6ICI4ZTEyZjRiNy05YzNhLTRmODMtOWI4ZC01MWEyYzdmNmU5ZDQiLA0KICAi
    c2lnbmVkX2hlYWRlcnMiOiBbDQogICAgew0KICAgICAgImRpZ2VzdCI6ICJTSEEtMjU2PTc5YTIwYTc0NDMzNjQyMDMwMTg
    zMDYwMGFkOWJkY2E5OTM1OTNmODc2MjA5YTAwNGI1OTliNTgzMDk1YjBhNjEiDQogICAgfSwNCiAgICB7DQogICAgICAiY2
    9udGVudC10eXBlIjogImFwcGxpY2F0aW9uL2pzb24iDQogICAgfQ0KICBdDQp9.DpuBNo2UgQhL7WLin4mpdZrbIpQq3tPv
    CX6HfktkxG7L5mk6a8OK1Hg0mQcZfFi3gelS-aL9kFS-6MoSy4csBg">EXAMPLE
136               required: true
137               schema:
138                 type: string
139             Digest:
140               description: Digest of the message payload, to comply with the INTEGRITY_REST_02
    security pattern. According to RFC 3230 Section 4.2 <a target="blank" href="https://www.rfc-edi
    tor.org/rfc/rfc3230.html#section-4.2">RFC 3230 §4.2</a>, the format MUST be the following diges
    t-algorithm=encoded digest output.
141               required: true
142               schema:
143                 type: string
144                 example: SHA-256=79a20a744336420301830600ad9bdca993593f876209a004b599b583095b0a
    61
145             Cache-Control:
146               $ref: "#/components/headers/CacheControlHeader"
147             RateLimit-Limit:
148               $ref: "#/components/headers/RateLimitLimitHeader"
149             RateLimit-Remaining:
150               $ref: "#/components/headers/RateLimitRemainingHeader"
151             RateLimit-Reset:
152               $ref: "#/components/headers/RateLimitResetHeader"
153           content:
154             application/json:
```

```yaml
            schema:
              $ref: "#/components/schemas/EServiceResponse"
      "400":
        description: Bad Request
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/ProblemDetails"
        headers:
          RateLimit-Limit:
            $ref: "#/components/headers/RateLimitLimitHeader"
          RateLimit-Remaining:
            $ref: "#/components/headers/RateLimitRemainingHeader"
          RateLimit-Reset:
            $ref: "#/components/headers/RateLimitResetHeader"
      "401":
        description: Unauthorized
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/ProblemDetails"
        headers:
          RateLimit-Limit:
            $ref: "#/components/headers/RateLimitLimitHeader"
          RateLimit-Remaining:
            $ref: "#/components/headers/RateLimitRemainingHeader"
          RateLimit-Reset:
            $ref: "#/components/headers/RateLimitResetHeader"
          WWW-Authenticate:
            $ref: "#/components/headers/WWWAuthenticateHeader"
      "404":
        description: Wallet Instance identifiers not found
        content:
          application/problem+json:
            schema:
              $ref: "#/components/schemas/ProblemDetails"
        headers:
          RateLimit-Limit:
            $ref: "#/components/headers/RateLimitLimitHeader"
          RateLimit-Remaining:
            $ref: "#/components/headers/RateLimitRemainingHeader"
          RateLimit-Reset:
            $ref: "#/components/headers/RateLimitResetHeader"
      "429":
        description: Too Many Requests
        content:
          application/problem+json:
```

```yaml
              schema:
                $ref: "#/components/schemas/ProblemDetails"
            headers:
              RateLimit-Limit:
                $ref: "#/components/headers/RateLimitLimitHeader"
              RateLimit-Remaining:
                $ref: "#/components/headers/RateLimitRemainingHeader"
              RateLimit-Reset:
                $ref: "#/components/headers/RateLimitResetHeader"
          "500":
            description: Internal Server Error.
            content:
              application/problem+json:
                schema:
                  $ref: "#/components/schemas/ProblemDetails"
            headers:
              Retry-After:
                $ref: "#/components/headers/RetryAfterHeader"
          "503":
            description: Service Unavailable
            content:
              application/problem+json:
                schema:
                  $ref: "#/components/schemas/ProblemDetails"
            headers:
              Retry-After:
                $ref: "#/components/headers/RetryAfterHeader"

tags:
  - name: status
    description: Endpoint di health check dell'API.
  - name: instance
    description: Retrieve information about the user's death.

components:
  securitySchemes:
    DPoPVoucherAuth:
      type: http
      scheme: DPoP
      description: >-
        PDND Voucher issued with DPoP binding (POP_DPoP, RFC 9449). The
        `Authorization` header MUST be `DPoP <voucher>` (`token_type` is `DPoP`).
        This scheme MUST be used together with `DPoPProof`. It MUST NOT be
        modelled as HTTP Bearer: a conformant Provider rejects
        `Authorization: Bearer` for a DPoP-bound Voucher.
    DPoPProof:
      type: apiKey
```

```yaml
      in: header
      name: DPoP
      description: >-
        DPoP proof JWT (RFC 9449). REQUIRED together with `DPoPVoucherAuth`
        (POP_DPoP). MUST NOT be sent with `BearerAuth` (POP_TPoP), where proof
        of possession is `Agid-JWT-TrackingEvidence`.
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: >-
        PDND Voucher issued as Bearer (POP_TPoP). The `Authorization` header
        MUST be `Bearer <voucher>` (`token_type` is `Bearer`). This scheme MUST
        be used together with `TrackingEvidenceAuth` as proof of possession.
        This is REST_JWS_2021_POP via TrackingEvidence, not REST_JWS_2021_Bearer.
    TrackingEvidenceAuth:
      type: apiKey
      in: header
      name: Agid-JWT-TrackingEvidence
      description: >-
        TrackingEvidence JWT. REQUIRED together with `BearerAuth` (POP_TPoP) as
        proof of possession of the PDND Voucher. The same header is also an
        optional operation parameter, so it MAY be sent with the POP_DPoP pair
        (`DPoPVoucherAuth` + `DPoPProof`) when the Provider requires
        AUDIT_REST_02. The DPoP proof remains the proof of possession.
        See e-Service PDND in the Technical Specifications.

  headers:
    CacheControlHeader:
      schema:
        type: string
        enum:
          - no-store
      description: no-store
    RateLimitLimitHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: Maximum number of requests within the time window.
    RateLimitRemainingHeader:
      schema:
        type: integer
        format: int32
        minimum: 0
      description: Remaining requests within the time window.
    RateLimitResetHeader:
```

```
296       schema:
297         type: integer
298         format: int32
299         minimum: 0
300       description: UTC epoch in seconds, corresponding to when the window for the current rate
    limit will reset.
301     RetryAfterHeader:
302       schema:
303         type: integer
304         format: int32
305         minimum: 0
306       description: Seconds to wait before receiving another response.
307     WWWAuthenticateHeader:
308       schema:
309         type: string
310         example: >-
311           Bearer error="invalid_token", error_description="The access token expired"
312       description: The request cannot be fulfilled because the Voucher is expired, revoked or o
    therwise malformed. See <a target="blank" href="https://datatracker.ietf.org/doc/html/rfc6750.h
    tml#section-3">RFC6750</a> and <a target="blank" href="https://datatracker.ietf.org/doc/html/rf
    c9449.html#section-7.1-11">RFC9449</a> for details.
313
314   schemas:
315     EServiceResponse:
316       properties:
317         result:
318           type: object
319           properties:
320             revoked:
321               type: array
322               items:
323                 type: string
324               description: List of Wallet Instances successfully rekoved.
325             not_found:
326               type: array
327               items:
328                 type: string
329               description: List of Wallet Instances not revoked as their identifier were not fo
    und at the Wallet Provider.
330             already_revoked:
331               type: array
332               items:
333                 type: string
334               description: List of Wallet Instances that were already revoked by the Wallet Pro
    vider.
335           required: [revoked, not_found, already_revoked]
336           description: >-
```

```
337             JSON object specifying which Wallet Instance was successfully revoked, which was al
    ready revoked and which was not found.
338         result_description:
339           description: Response Description.
340           type: string
341           example: The service partially process the revocation of the requested Wallet Instanc
    es as some identifiers were not found.
342           required: [result, result_description]
343     ProblemDetails:
344       type: object
345       description: RFC7807-compliant problem details object for error responses.
346       properties:
347         type:
348           type: string
349           format: uri
350           description: An absolute URI that identifies the problem type.
351         title:
352           type: string
353           description: A short, human-readable summary of the problem type.
354         status:
355           type: integer
356           format: int32
357           description: The HTTP status code generated by the origin server for this occurrence
    of the problem.
358         detail:
359           type: string
360           description: A human-readable explanation specific to this occurrence of the problem.
361         instance:
362           type: string
363           format: uri
364           description: An absolute URI that identifies the specific occurrence of the problem.
365       required: [title, status, detail]
366     NotifyUserDeath:
367       required:
368         - wallet_instance_ids
369       type: object
370       properties:
371         wallet_instance_ids:
372           type: array
373           items:
374             type: string
375           description: >-
376             Identifiers of the Wallet Instances whose PID was revoked due to User death.
```

Indice

A

Access CA, **675**
Access Certificate, **677**
Access Certificate Authority, **675**
App di Verifica, **681**
AS Registry, **684**
Asset Critici, **675**
Associazione Crittografica con l'Utente, **676**
Attestato, **676**
Attestato Elettronico, **676**
Attestato Elettronico di Attributi, **676**
Attestato Elettronico di Attributi Qualificati, **676**
Attestato Elettronico di Attributi rilasciato da o per conto di un ente pubblico, **676**
Attestato Elettronico di Dati di Identificazione Personale di ambito nazionale, **681**
Attestato Elettronico di Interesse Pubblico, **676**
Attestato Elettronico Pubblico di Attributi, **676**
Attori Primari, **676**
Attributi, **676**
Attributi dell'Utente, **676**
Autenticazione, **676**
Authentic Source Registry, **684**
Authentication Button, **686**
Autorità di Federazione, **677**

B

Backend del Fornitore di Wallet, **677**
Backend della Relying Party, **677**
Brand Identity, **686**
Brand Manual, **687**

C

Call To Action, **687**
Catalogo, **687**
Catalogo degli Attestati Elettronici, **677**
Certificate Signing Request, **677**
Certificato di Accesso, **677**
Certificato di Registrazione, **677**
Credential Issuer, **678**
Cryptographic Hardware Key Tag, **677**
Cryptographic Hardware Keys, **677**

D

Dataset_id, **677**
Dati di Identificazione Personale, **678**
Discovery Page, **687**
Divulgazione Selettiva, **678**

E

EAA Provider, **679**
eIDAS Trusted List, **678**
Engagement Button, **687**
Enti Nazionali di Accreditamento, **678**
Entità Organizzativa, **678**

Esperienza Utente, **687**

F

Federation Registry, **678**
Fonte Autentica, **678**
Fornitore di Attestati Elettronici, **678**
Fornitore di Attestati Elettronici di Attributi, **679**
Fornitore di Attestati Elettronici di Attributi Qualificati, **679**
Fornitore di Attestati Elettronici di Dati di Identificazione Personale, **679**
Fornitore di Certificati di Registrazione, **679**
Fornitore di Servizi, **679**
Fornitore di Wallet, **679**
Fornitore Qualificato di Firme Elettroniche, **679**

G

Gestore di Identità Digitale, **679**

H

Holder, **685**
Holder Key Binding, **676**

I

Identity and Access Management, **679**
Identity Matching, **680**
Identity Reconciliation, **680**
Interfaccia, **687**
Intermediario, **680**
Intermediario di Federazione, **680**
Intermediario di Relying Party, **680**
Intermediario IT-Wallet, **680**
Istanza del Wallet, **681**
Istanza di Relying Party, **681**
IT-Wallet ID, **681**

K

KA, **681**
Key Attestation, **681**
Key Attestation APIs (OEM), **681**
Keystore, **682**
Keystore hardware-backed interno locale, **682**

L

Livello di Garanzia, **682**

M

Member State Trusted List Provider, **682**
Metadata, **682**
Modello di Interazione, **687**
Modello di Servizio, **687**
MS TLP, **682**

N

National Trust Anchor, **682**

O

Object_id, **682**
Organismo di Valutazione della Conformità, **683**

P

Person Identification Data, **678**
PID, **678**
PID Provider, **679**
Policy Language, **683**
Processo di Accreditamento, **683**
Processo di Certificazione, **683**
Processo di Notifica, **683**
Processo di Registrazione, **683**
Processo di Supervisione, **683**
Provider of Person Identification Data, **679**
Provider of Registration Certificates, **679**
Pseudonimo, **683**

Q

QEAA Provider, **679**

R

Reg. Cert. Provider, **679**
Registrar, **683**
Registration Authority, **683**

Registration Certificate, **677**
Registro del Sistema IT-Wallet, **683**
Registro delle Fonti Autentiche, **684**
Relazione di Fiducia, **684**
Relying Party, **679**
RFC
RFC 1950, 435, 442, 859
RFC 1951, 435, 442, 859
RFC 2511, 632
RFC 3230, 744, 746
RFC 3280, 283
RFC 3339, 402
RFC 3986, 439
RFC 4122, 600, 608
RFC 4648, 805
RFC 5280, 206, 381, 437, 580, 587, 693
RFC 5646, 358, 359
RFC 5869, 649
RFC 6749, 447, 456, 457, 523, 595, 597, 606–609, 727–729, 792, 835, 836
RFC 67491, 607
RFC 6749#section-4.1.2, 604
RFC 6749#section-4.1.2.1, 524, 537, 538, 601, 605, 627
RFC 6749#section-5.2, 601, 609, 610, 731, 732, 737
RFC 6750, 617, 622, 839
RFC 6750#section-3, 737, 747
RFC 6750#section-3.1, 737
RFC 7231, 574, 788
RFC 7515, 206, 207, 356, 381, 383, 437, 525, 580, 587, 608, 611–615, 728–731, 744, 745
RFC 7516, 648, 650
RFC 7516#section-4.1.1, 323, 525, 577, 580, 584, 587, 596, 599
RFC 7517, 127, 437, 525, 587, 608, 615, 733–735, 737, 779, 780
RFC 7518, 647–651
RFC 7519, 127, 382, 437, 438, 441, 453, 525, 531, 577, 580, 581, 584, 587, 596, 597, 599, 600, 605, 608, 610–613, 615, 616, 728, 730, 745–747
RFC 7521, 604, 727, 728
RFC 7523, 727, 728
RFC 7591#section-2, 375
RFC 7591#section-3.2.1, 375
RFC 7636, 356, 447, 451, 457, 770, 791, 836
RFC 7636#section-4.2, 597
RFC 7636#section-4.3, 597
RFC 7638, 611, 613, 731
RFC 7638#section_3, 437, 577, 580, 584, 587, 596, 611, 612
RFC 7800, 577, 581, 584
RFC 7807, 574, 788
RFC 8259, 600, 832
RFC 8414, 356, 358, 595
RFC 8414#section-2, 356, 357
RFC 8725, 531, 596, 608
RFC 8949, 389, 411, 847
RFC 8963, 730
RFC 9027, 456
RFC 9052, 390, 848
RFC 9053, 390
RFC 9068, 611–613, 730
RFC 9101, 357, 447, 452, 455, 525, 532, 596, 827–829
RFC 9110, 441, 608, 619, 621, 861
RFC 9126, 447, 451–453, 455, 456, 577, 581, 584, 587, 596, 597, 599, 600, 604, 791, 827–829, 833
RFC 9126#as_metadata, 356
RFC 9126#section-2.3, 600, 601
RFC 9207, 605, 793
RFC 9360, 390
RFC 9396, 447, 452, 598, 792
RFC 9449, 357, 457, 460, 465, 472, 473, 607, 610–614, 618, 716, 719, 727, 729, 731, 736, 739, 741, 744, 836, 838, 840
RFC 9449#section-4, 793, 794
RFC 9449#section-5, 732
RFC 9449#section-7.1, 747

RFC 9457, 747
RFC 9864, 647

S

Segnale (*Signal Hub*), **684**
Selection Page, **687**
Self-Sovereign Identity, **684**
Servizio di Integrità del Dispositivo, **684**
Signal Hub, **684**
Sistema IT-Wallet, **684**
Soggetto Aggregatore, **680**
Soluzione di Relying Party, **684**
Soluzione Wallet, **684**
Soluzioni Tecniche, **684**
Specifiche Tecniche, **685**

T

Titolare, **685**
Touchpoint, **688**
Trust, **685**
Trust Attestation, **685**
Trust Evaluation, **685**
Trust Framework, **685**
Trust Layer, **685**
Trust Mark, **688**
Trust Model, **685**
Trust Relationship, **684**
Trusted List, **685**

U

Utente, **685**

V

Verificatore di Attestati Elettronici, **685**
Verificatore di Credenziali, **685**
Vista di Dettaglio, **688**
Vista in Anteprima, **688**
Visual Identity, **688**

W

Wallet Instance Attestation, **686**
Wallet Provider Backend, **677**
Wallet Secure Cryptographic Application, **686**
Wallet Secure Cryptographic Device, **686**
Wallet Unit, **686**
Wallet Unit Attestation, **686**
Wallet-Relying Party, **679**
WIA, **686**
WUA, **686**