WE BUILD
CONSORTIUM

Co-funded by
the European Union

WE BUILD — Architecture & Integration Blueprint (D4.1)

Table of Contents

1. Executive Summary and Project Context

1.1 Background

WE BUILD is a Large Scale Pilot (LSP) funded by the European Commission. The project tests how the European Digital Identity (EUDI) Wallet and the European Business Wallet (EBW) can support cross-border business processes across the EU.

The goal is practical: reduce administrative barriers that slow down companies when they operate across borders, such as opening a bank account, exchanging trusted business documents, or registering a branch in another country.

WE BUILD is organised around 13 concrete use cases that demonstrate how digital identity wallets can support real business processes. These use cases are grouped into three main areas:

- Business (WP2) — processes such as company registration, mandates and representation.
- Supply Chain (WP2) — logistics, transport documentation and electronic invoicing.
- Payments & Banking (WP3) — secure payments and simplified onboarding to financial services.

1.2 WE BUILD's Role in the EUDI and EBW Journey

WE BUILD operates within the emerging EUDI and EBW ecosystem, but it is not the final production environment. Instead, the project acts as a large-scale pilot where technical solutions, governance models and interoperability rules can be tested through real use cases.

In the final EUDI ecosystem, every EU citizen will receive an EUDI Wallet at Level of Assurance (LoA) High.

WE BUILD focuses in particular on the EBW, designed for economic operators to manage mandates, exchange trusted business documents such as electronic invoices, and receive legally valid notifications. Some EBW functions, such as onboarding and data portability, will operate at LoA Substantial.

1.2.1 Bridging ARF Gaps through Specifications and Testing

The future EUDI ecosystem is defined through the Architecture Reference Framework (ARF). Because the ARF is still evolving, it does not yet cover every implementation detail. In the WE BUILD pilot environment, the full certification and qualification schemes used in production cannot always be applied.

To address these gaps, WE BUILD defines project-specific implementation rules through:

- WE BUILD Conformance Specifications (WBCS) — technical rules that implementations must follow.
- Architectural Decision Records (ADRs) — documented architecture decisions that guide the project.

In the final ecosystem, wallets and services must undergo formal certification by national supervisory bodies.

WE BUILD does not	WE BUILD does
certify EUDI wallets	provide WE BUILD wallets that pass ITB testing
rely on eIDAS-eID	provide WE BUILD eID with fictitious but realistic identities
create eIDAS-qualified e-signatures	define WE BUILD qualification of e-signatures focused on technical interoperability
use real MS registries	use real public sector bodies where possible, otherwise simulate them using fictitious
use the EC List of Trusted Lists (LoTL)	provide a WE BUILD LoTL
use the MS Trusted lists (TL)	provide WE BUILD TLs with input from supervisory bodies where available
reach production-level legal liability	operate within the WE BUILD agreement and trust framework, not within eIDAS
deal with national policy-making	use the WP5 MS Forum to indicate alignment with national policy-making
deal with universal definitions	define WE BUILD semantics within the available timeframe
issue EUDIW-RP access certificates	issue WE BUILD RP access certificates
issue eIDAS-QEAA	define WE BUILD QEAA focused on technical interoperability

1.3 Work Package 4 (WP4) - General Capabilities

The technical groups in WP4 - Architecture, Semantics, Wallet Providers, PID & EBWOID Providers, Qualified Trust Service Provider (QTSP), Trust Registry Infrastructure, and Test Infrastructure - provide the technical capabilities that support the use cases.

WP2 and WP3 use cases are expected to use the capabilities provided by WP4 rather than developing parallel technical solutions.

To ensure interoperability across participants, WE BUILD uses three levels of documentation:

1. This **Architecture & Integration Blueprint (D4.1)** - the high-level architecture and system overview.
2. **Architectural Decision Records (ADR)** - explains major architecture decisions and the reasoning behind them.
3. **WE BUILD Conformance Specifications (WBCS)** — defines the detailed technical requirements that implementations must follow.

The governance process behind ADRs and WBCS, including how decisions are proposed and

adopted, is described in Chapter 7.

The Interoperability Testbed (ITB) is a first step toward understanding conformity assessment requirements. In a controlled consortium environment, regulatory and technical specifications are translated into executable interoperability scenarios.

1.4 Getting Started

The Blueprint is the starting point for understanding how WE BUILD works.

- Technical teams should begin with the WBCS to implement their interfaces.
- Architects should review the ADRs to understand the key architecture decisions.
- Every implementation must pass through the Interoperability Testbed (ITB) before participating in pilots.

2. Regulatory and Foundational Alignment

The WE BUILD architecture aligns with two key regulatory instruments: Regulation (EU) No 910/2014, as amended by Regulation (EU) 2024/1183 (commonly referred to as the amended eIDAS Regulation), and the proposed European Business Wallet proposal for economic operators.

2.1 The EUDI Framework

WE BUILD aligns with the legal and technical framework for EUDI wallets. Users can authenticate and present identity and attribute information while retaining control over what data is shared through selective disclosure and explicit consent.

The amended eIDAS Regulation is supported by several implementing acts defining the technical and governance framework for the EUDI Wallet ecosystem, most importantly:

Core Wallet Architecture and Technical framework

- 2024/2979 — Integrity and core functionalities
- 2024/2982 — Protocols and interfaces
- 2024/2981 — Certification framework
- 2024/2980 — Notification obligations within the Wallet ecosystem

Person Identification Data (PID) and Electronic Attestations of Attributes (EAA)

- 2024/2977 — PID and electronic attestations of attributes
- 2025/1569 — Electronic attestations of attributes

Wallet Ecosystem Governance and Relying Parties

- 2025/848 — Registration of wallet-relying parties
- 2025/849 — Submission of information on certified European Digital Identity Wallets

- 2025/847 — Reactions to Wallet security breaches
- 2025/846 — Cross-border identity matching for natural persons

Trust Services and QSCD Framework

- 2025/1566 — Verification of identity and attributes (QTSs)
- 2025/1567 — rQSCD management
- 2025/1570 — Notification of certified or cancelled QSCDs
- 2025/1572 — QTSP initiation, notification and verification

2.1.1 Standardisation and Technical Specifications

The European Commission, together with the European Digital Identity Cooperation Group, has published:

- The ARF specifies main functionalities, roles and responsibilities, architecture and design principles, attestation formats and protocols, trust model, certification, and risk management of the EUDI Wallet ecosystem.
- The Technical Specifications specify more technical details of selected topics derived from the ARF. The technical specifications describe various topics such as Relying Party registration, zero-knowledge proofs, attestation rulebooks, schemas and catalogues.

Furthermore, there are several standardisation organisations that contribute with standards for the EUDIW ecosystem.

- **ETSI ESI:** ETSI ESI is a European Standardisation Organisation (ESO) that creates technical standards and European Norms for electronic identity and signatures supporting the eIDAS regulation. ETSI ESI has published approximately 80 standards for QTSP conformity assessment, protocols and formats for digital signatures, as well as protocols and formats for the EUDI Wallet. ETSI ESI has received the standardisation request STF 705 from the EU Commission to create and/or update several standards for the EUDI Wallet ecosystem.
- **CEN TC224:** CEN Technical Committee 224 (TC224) is an ESO that has published several standards related to identification and devices with secure elements. More specifically, CEN TC224 WG17 are standardizing Common Criteria protection profiles of QSCD/WSCA, CEN TC224 WG18 develop standards related to biometric solutions, whilst CEN TC224 WG20 are creating standards related to EUDI Wallet onboarding and access control.
- **ISO/IEC:** ISO is an international standardisation organisation and the International Electrotechnical Commission (IEC) develops international standards for electronic technologies. The international standardisation activities related to digital identities are performed within ISO/IEC Joint Technical Committee (JTC) 1 "Information Security". More specifically, several ISO/IEC standards are applicable to Common Criteria certification, conformity assessment and evaluation of the EUDI Wallet solutions. Furthermore, ISO/IEC has standardised the mobile driving license (ISO mDL) in ISO 18013-5, which is a PID format for the EUDI Wallet.
- **IETF:** The Internet Engineering Task Force (IETF) creates technical standards that comprise the internet protocol suites. More specifically, IETF PKIX covers secure data exchanges and formats in the area of electronic signatures, PKI and trust services. Most notably, IETF has published

standards for PKIX X.509 certificate and CRL profiles, OCSP, TLS and SD-JWT, which are relevant for the EUDI Wallet ecosystem. Furthermore, some of the IETF standards are used as basis by ETSI ESI, which have created European profiles of Qualified Certificates, AdES signature formats, SD-JWT VC, etc.

- **OpenID Foundation:** The OpenID Foundation is an industrial standardisation organisation that develops open standards for identity, federation and security. The following OpenID standards are relevant for the EUDIW technical architecture: OpenID Connect Core (OIDC), OpenID For Verifiable Credential Issuance (OID4VCI), OpenID For Verifiable Presentations (OID4VP), and OpenID High Assurance Interoperability Profile (HAIP). OID4VP, OID4VCI and HAIP are used as the foundation for the ETSI TS 119 472 standardisation of EUDI Wallet protocols.
- **W3C:** The World Wide Web Consortium (W3C) is an international standardisation organisation. The following W3C standards are relevant to the EUDI Wallet technical architecture: W3C Verifiable Credentials Data Model, W3C Web Authentication (WebAuthn), and W3C Digital Credentials API. More specifically, the W3C Verifiable Credentials Data Model is referenced as basis for an ETSI TS 119 472 EAA profile.
- **Cloud Signature Consortium (CSC):** The Cloud Signature Consortium (CSC) is an international standardisation organisation focusing on compliant digital signature creation in the cloud. The CSC specification "CSC API v2 - Architectures and protocols for remote signature applications" is referenced by the EUDI Wallet architecture and is used as basis for the ETSI TS 119 432 standard.

In addition to the aforementioned standardisation organisations, the European Cybersecurity Agency (ENISA) is developing the EUDI Wallet Certification Scheme, which will be published as an implementing regulation under the Cybersecurity Act. The purpose of the EUDI Wallet Certification Scheme is to harmonise the national certifications of the EU Member States' EUDI Wallets.

2.2 The EBW Framework

The EBW framework is introduced through the European Commission’s Digital Package proposal as part of its 2025 Work Programme. The proposal aims to establish the EBW as a harmonised digital solution that reduces administrative burden and allows companies and public authorities to identify, authenticate and exchange data with legal effect across the European Union.

The EBW framework complements the EUDI framework by addressing the needs of economic operators and public authorities. It supports the digital management of representation rights and mandates, provides a secure channel for exchanging official documents and attestations, and includes support for a common directory. Interoperability with the EUDI Wallet is a core requirement.

The proposal supports the management and use of EAA, including owner identification data with selective disclosure. It defines requirements for authenticating owners and authorised users through (Q)EAAs and enables links between EAAs and other attestations. Access to EAAs by relying parties requires proper authorisation.

The framework relies on existing eIDAS trust services such as qualified electronic signatures, seals, timestamps and registered delivery services.

The proposal also introduces a European Digital Directory maintained by the Commission. The directory functions as a trusted internal system where EBW providers notify relevant service information and where digital addressing can be supported. Detailed requirements will be defined in future implementing acts.

The regulation supports role-based access so that multiple authorised users can operate a wallet. It also enables secure data exchange between EBWs, EUDI Wallets and relying parties, while allowing additional functionalities provided that core features remain unaffected.

From a technical perspective, the framework promotes the use of common protocols for sharing attestations. It requires secure onboarding using eID with a LoA of at least Substantial and mandates interoperability, secure communication interfaces, and mechanisms for validation and revocation. Further requirements will be defined in implementing acts.

Within WE BUILD, the proposed EBW framework is treated as a primary regulatory and architectural reference for business-focused identity and data exchange scenarios. Use case design and pilot activities align with the EBW model's legal structure, interoperability requirements and trust-service framework, while taking forthcoming implementing acts into account.

3. Architecture Overview

3.1 Architectural Principles

While the previous chapter describes the regulatory and architectural frameworks that WE BUILD aligns with, this chapter introduces the architectural principles guiding the design of the WE BUILD ecosystem.

- **Interoperability:** Wallet providers, issuers and verifiers interact across organisational and national boundaries.
- **Reusability:** The architecture builds on existing EU digital infrastructure and results from previous Large Scale Pilots.
- **Security by design:** Security controls are integrated into the architecture from the start.
- **Privacy by design:** Users retain control over personal and organisational data through selective disclosure and explicit consent.

3.2 The Ecosystem at a Glance

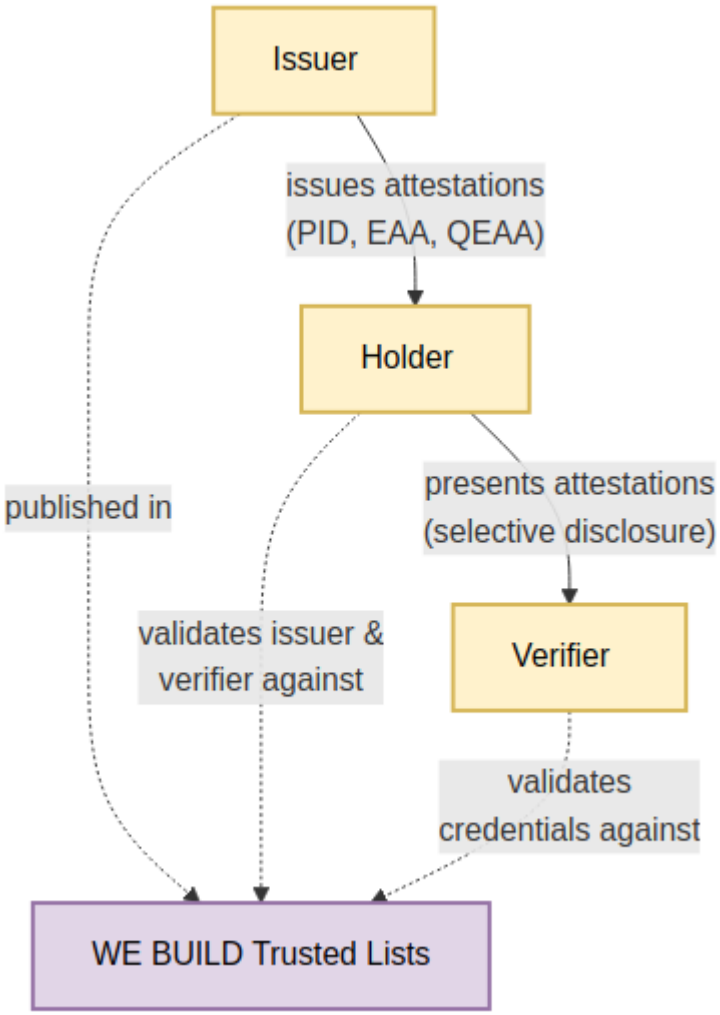
The EUDI Wallet and EBW ecosystem follows the common three-party attestation model. In this model, three primary actors interact: issuer, holder and verifier. A trust framework supports these actors by providing the trust anchors used for validation.

1. **Holder** — the wallet controlled by a natural or legal person.
2. **Issuer** — an entity that issues attestations to the Holder.
3. **Verifier** — a relying party that receives and validates attestations presented by the Holder.
4. **Trust framework** — the infrastructure used to validate trust relationships between ecosystem

participants (described in Chapter 6).

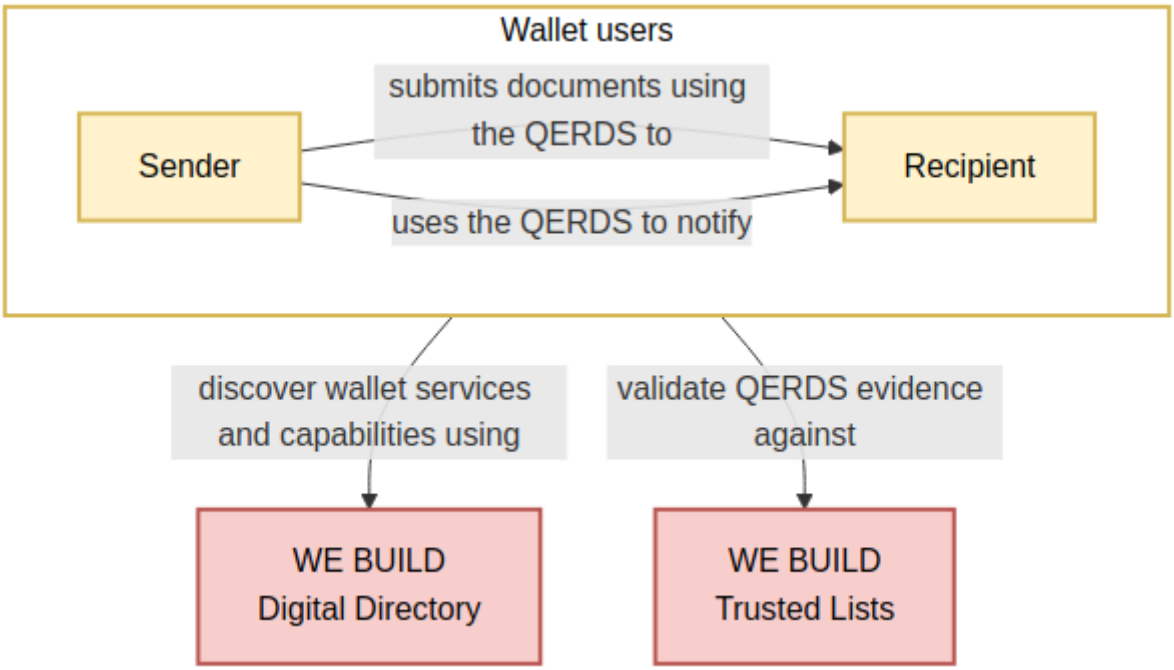
3.3 System Landscape

The diagram below illustrates the baseline trust topology of the EU wallet ecosystem. Issuers provide attestations to holders, holders present them to verifiers, and all actors validate trust relationships using the trusted lists. The trusted lists specify the recognised participants in schemes for electronic identification and trust services.



WE BUILD focuses primarily on wallets for economic operators and public sector bodies. In these scenarios, qualified electronic registered delivery services (QERDS) support trusted messaging between recognised participants. Accordingly, interactions between data senders and receipients may be routed through a Qualified Trust Service Provider (QTSP) providing a QERDS. The QTSP is recognised in a scheme for trust services, just like in the previous diagram, enabling other participants to verify QERDS evidence issued by the QTSP. The WE BUILD Digital Directory (simulating the European Digital Directory) provides economic and public sector bodies with digital addressing for secure routing of documents and notifications.

While this model can apply to any data transmission, the senders, recipients and their QERDS providers can take the issuer-holder-verifier roles as illustrated as above. The QERDS provides an additional layer in the WE BUILD ecosystem:



3.4 Wallet Types in WE BUILD

WE BUILD supports wallet solutions for both natural persons and economic operators.

Natural persons interact through EUDI Wallets, which enable individuals to authenticate and present personal identity attributes. Economic operators interact through EBW, which enable organisations to manage and present business-related attestations such as representation rights or organisational attributes.

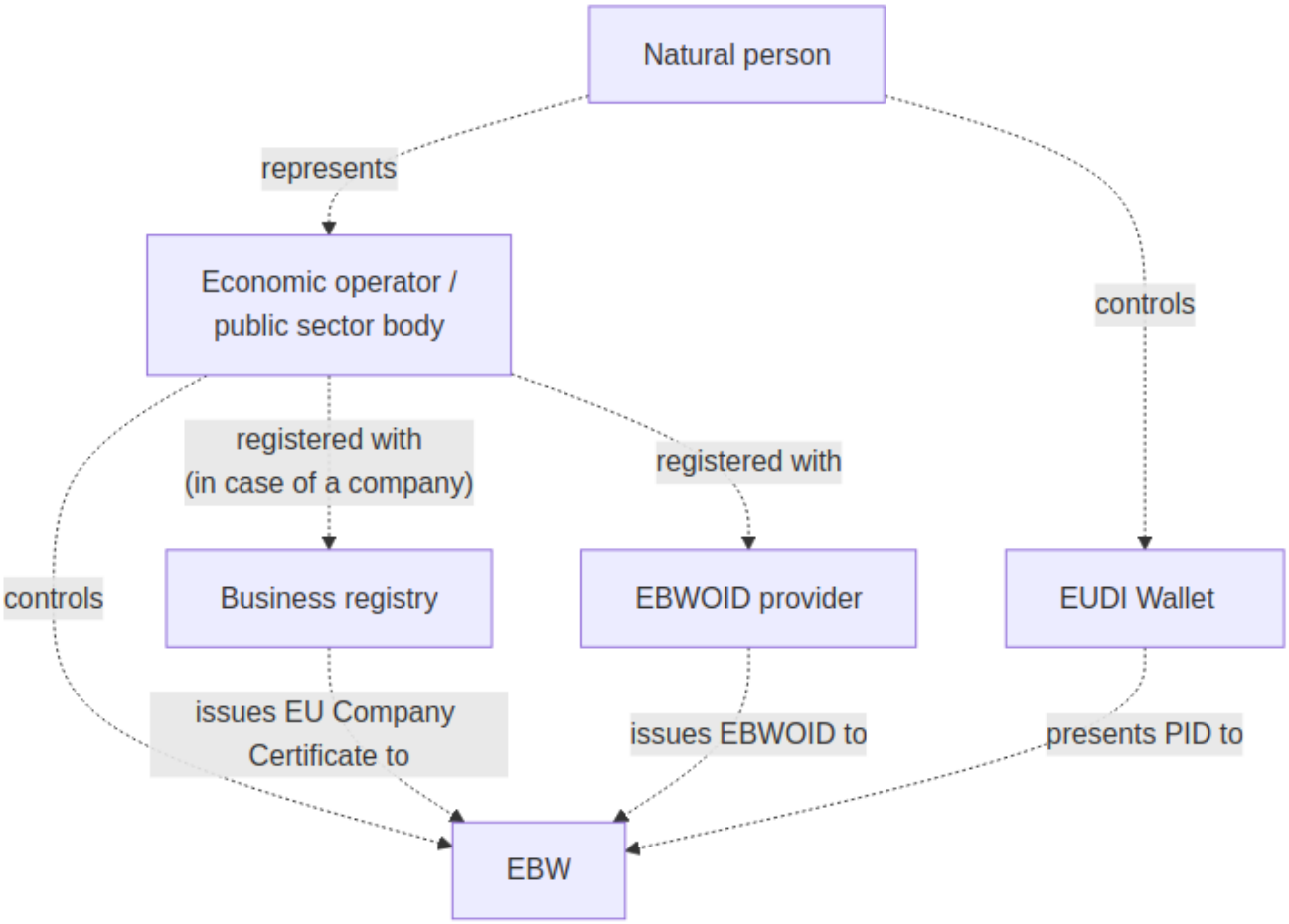
From a deployment perspective, wallet solutions can be implemented in several ways depending on the target users, operational requirements, and cryptographic architecture. In practice, three main implementation approaches are relevant within the WE BUILD ecosystem.

Wallet type	Typical context	Characteristics
Mobile wallets (on-device)	Natural persons	Wallet application running on a user's smartphone, with credentials stored and used locally on the device.
Server or Web-based wallets	Economic operators	Wallet services operated in backend infrastructure and accessed through Web interfaces or enterprise systems.
Hybrid wallets	Both contexts	Combine device-based interaction with backend cryptographic infrastructure.

The underlying cryptographic architecture of wallets is defined in the ARF and related standards. This Blueprint therefore focuses on the interactions and interoperability patterns relevant for WE

BUILD rather than repeating the detailed wallet architecture definitions.

In practice, most deployments follow a mobile-first approach for natural persons and a server-based or enterprise-integrated approach for economic operators. Hybrid architectures may also be used to combine device-based user interaction with backend cryptographic services.



4. How the Wallet Interacts with Services

Chapter 3 introduced the main actors in the WE BUILD ecosystem. This chapter describes how these actors interact through wallet-based service flows.

4.1 Interaction Pattern: Attestation Issuance

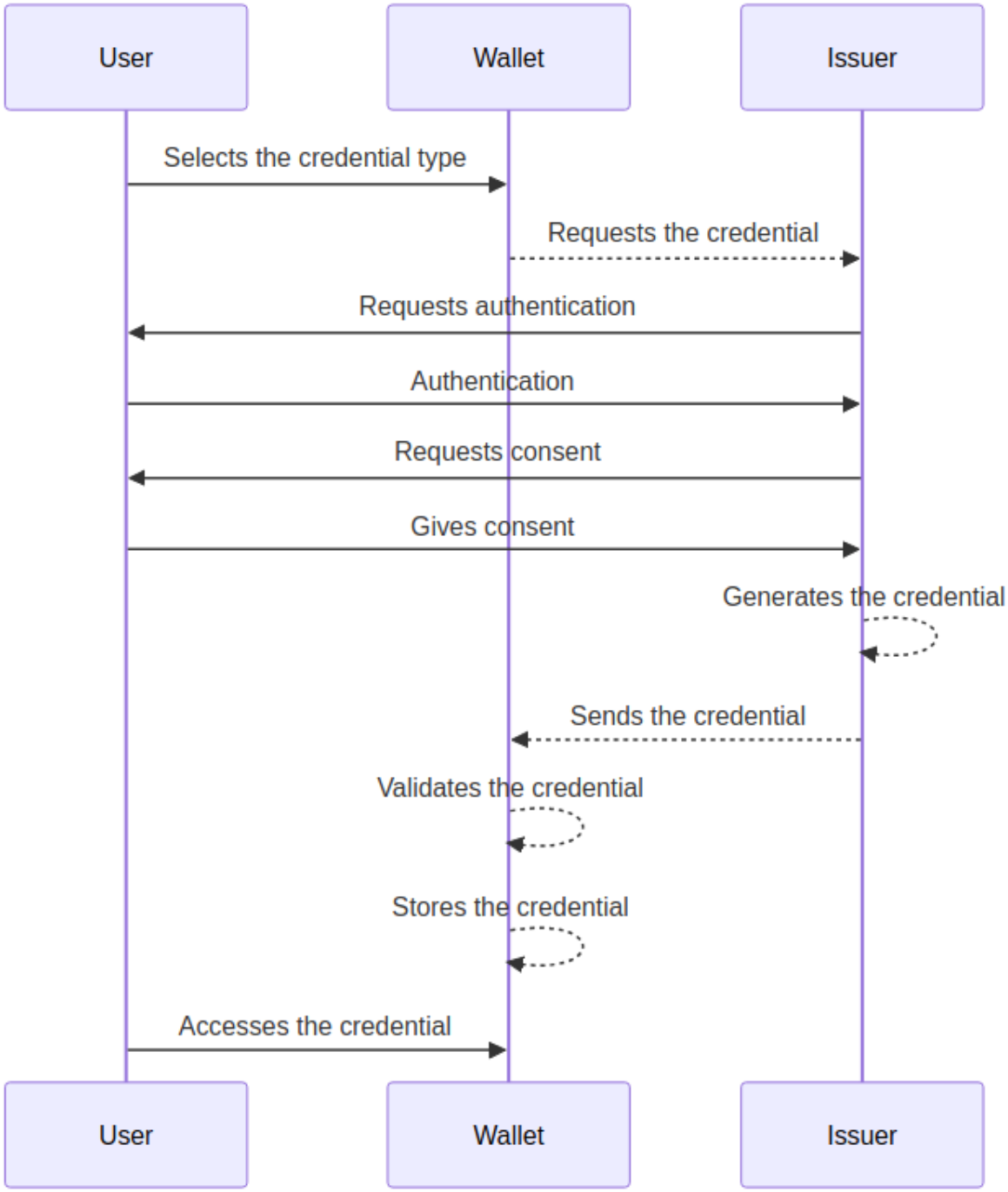
The WBCS for high-assurance credential issuance defines the requirements used in the project to ensure interoperable issuance of verifiable digital credentials between wallets and issuers. For reference on qualified electronic attestation of attributes, see the QEAA documentation.

The WE BUILD ecosystem mainly supports two credential issuance models, which differ in which actor initiates the process: wallet-initiated issuance and issuer-initiated issuance. If the credential cannot be issued immediately, deferred issuance is used. The wallet retries periodically until the credential is issued or an unrecoverable error occurs.

4.1.1 Wallet-initiated Issuance

This issuance flow is initiated by the user:

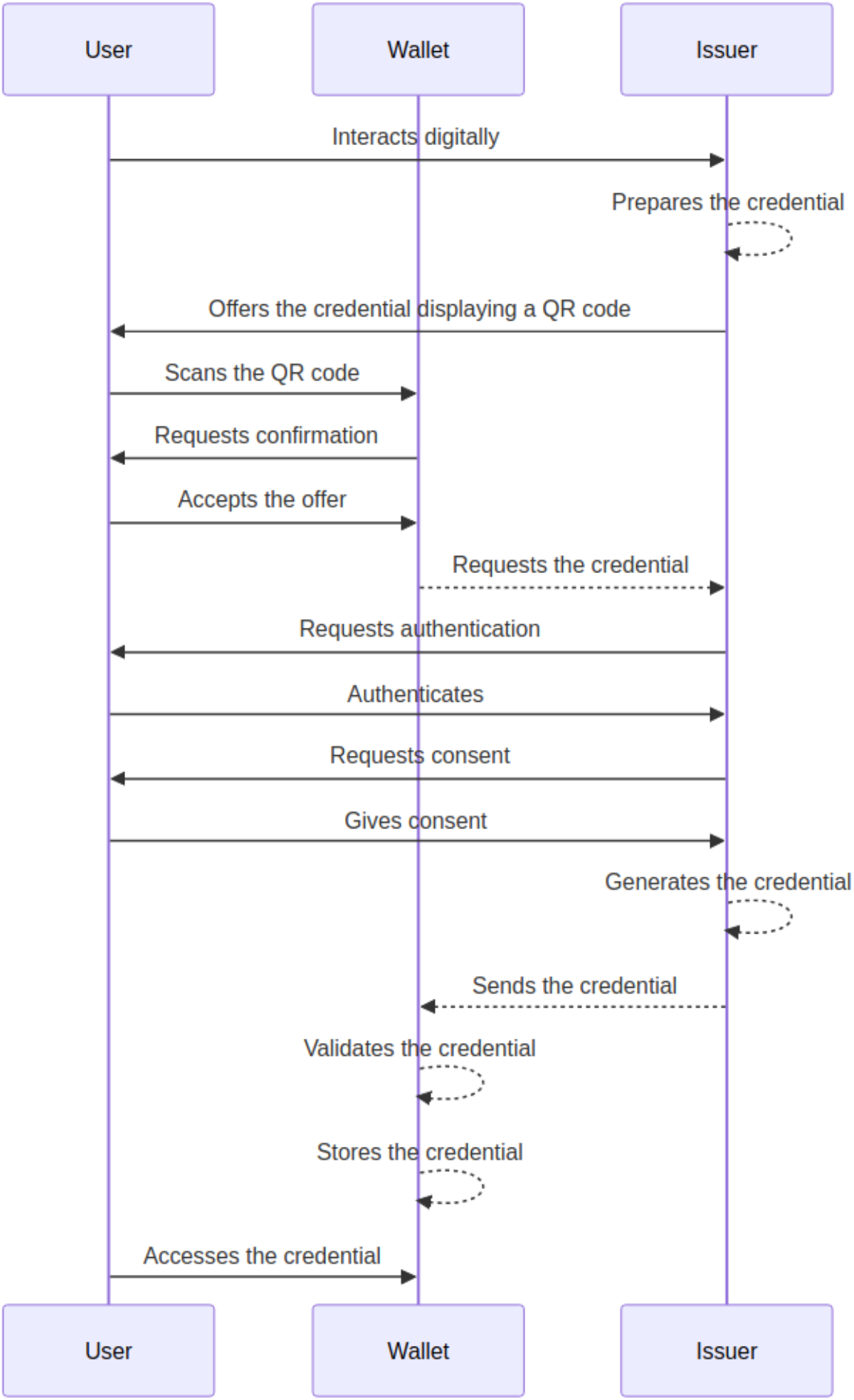
1. The user opens their wallet and selects the credential type to be issued (for example, a PID or a QEAA).
2. The wallet connects with the corresponding issuer and requests the credential.
3. The user authenticates with the issuer, following the procedure specified by the issuer itself.
4. The issuer requests the user's consent to issue the credential and send it to their wallet.
5. The issuer generates the credential and delivers it to the wallet.
6. The wallet verifies the authenticity of the credential and stores it. From this point, the user becomes responsible for managing the issued credential.



4.1.2 Issuer-initiated Issuance

This issuance flow is initiated by the issuer:

1. The user interacts with the issuer (for example, during a digital onboarding process).
2. The issuer prepares one or more credentials.
3. The issuer offers these credentials to the user. This can be done in several ways, both same-device and cross-device:
    - By displaying a QR code that the user shall scan with their wallet.
    - By sending a link to the wallet.
4. The wallet displays the offer and requests confirmation from the user.
5. The user authenticates with the issuer, following the procedure specified by the issuer itself.
6. The issuer requests the user's consent to issue the credential and send it to their wallet.
7. The issuer generates the credential and delivers it to the wallet.
8. The wallet verifies the authenticity of the credential and stores it. From this point, the user becomes responsible for managing the issued credential.


User
Wallet
Issuer
Interacts digitally
Prepares the credential
Offers the credential displaying a QR code
Scans the QR code
Requests confirmation
Accepts the offer
Requests the credential
Requests authentication
Authenticates
Requests consent
Gives consent
Generates the credential
Sends the credential
Validates the credential
Stores the credential
Accesses the credential
User
Wallet
Issuer

4.2 Interaction Pattern: Attestation Presentation (Receiving)

In this pattern, a verifier requests specific attestations from the wallet. The wallet presents the requested information, typically using selective disclosure mechanisms, and the verifier validates the received data.

The WE BUILD Conformance Specification for Credential Presentation describes how wallets and relying parties interoperate within the WE BUILD ecosystem. It covers presentation (request and response flows), interfaces between wallets and relying parties as well as security, privacy and interoperability requirements and same-device and cross-device invocation patterns.

4.3 Signature and Seal Integration

Wallets in WE BUILD provide the ability to create qualified electronic signatures and seals. This section describes the various integration models. For reference, see the QES documentation.

WE BUILD supports both wallet-centric and QTSP-operated approaches for electronic signatures and seals. In both models, the wallet provides the user interaction layer, while cryptographic operations may take place either locally or in remote infrastructure operated by a QTSP.

Both approaches are compatible with the architectural patterns described in the ARF. However, during the WE BUILD pilot phase not every wallet provider or QTSP is expected to implement every possible model. For interoperability across the consortium, WE BUILD therefore treats remote signing and sealing through QTSP-managed services with standardised interfaces as the common baseline. Local signing models may still be supported by individual wallet implementations, but they are not assumed as a uniform baseline for interoperability within the project.

This section aligns with the WP4 interoperability baselines defined for issuance and presentation flows. Proximity-based signing scenarios are currently outside the baseline protocol scope of the WE BUILD pilots.

In the WE BUILD pilot and ITB environment, eIDAS-qualified status cannot be achieved because the ITB operates outside the formal eIDAS certification framework. Any reference to “qualified” in WE BUILD therefore represents a technical demonstration only and does not constitute a legally valid qualified electronic signature. The prerequisites for eIDAS-qualified status remain unchanged, including use of a Qualified Signature Creation Device (QSCD) and a qualified certificate issued by a QTSP that is listed on an official national Trusted List.

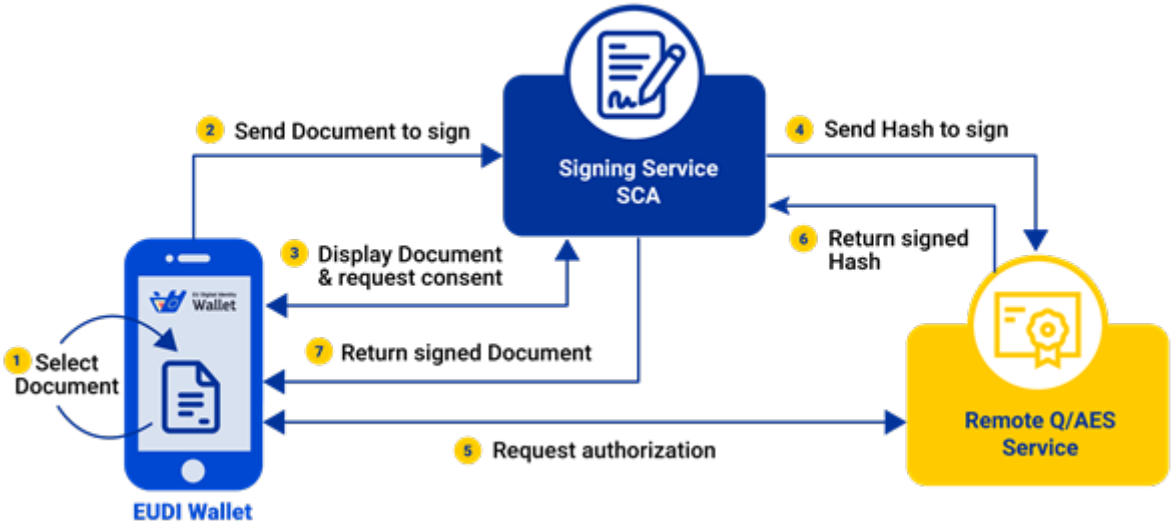
4.3.1 Wallet-centric Signing Model

In the wallet-centric model, the EUDI Wallet is the central component of the electronic signature process. Three distinct signing processes are considered, depending on where the Signature Creation Application (SCA) runs and where the Signature Creation Device (SCD) is hosted.

4.3.1.1 1) Remote Signing with External SCA

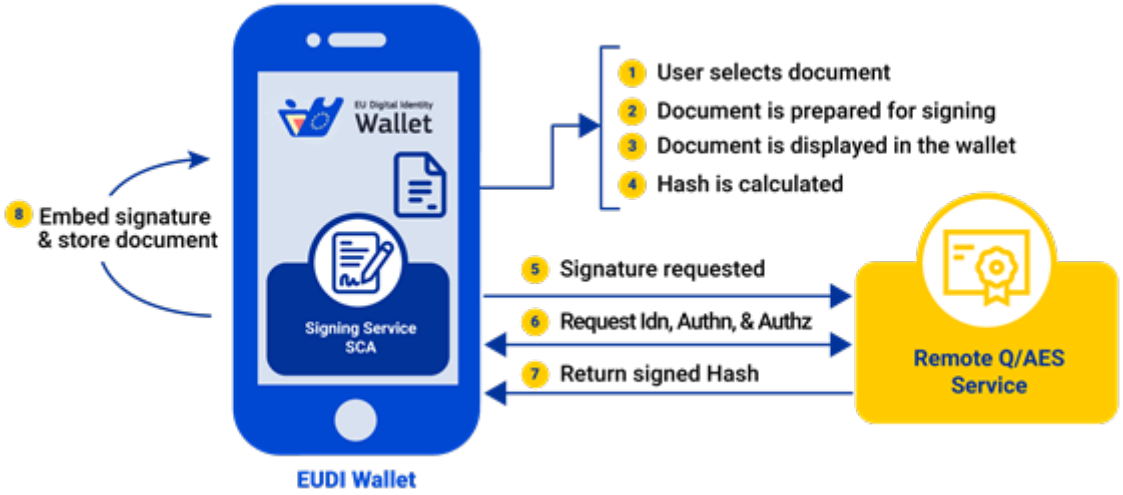
The user initiates signing from the wallet, while the SCA is external. The document is sent to the

external SCA for review and consent, after which the signing request is forwarded to a remote SCD that creates the signature and returns the result.



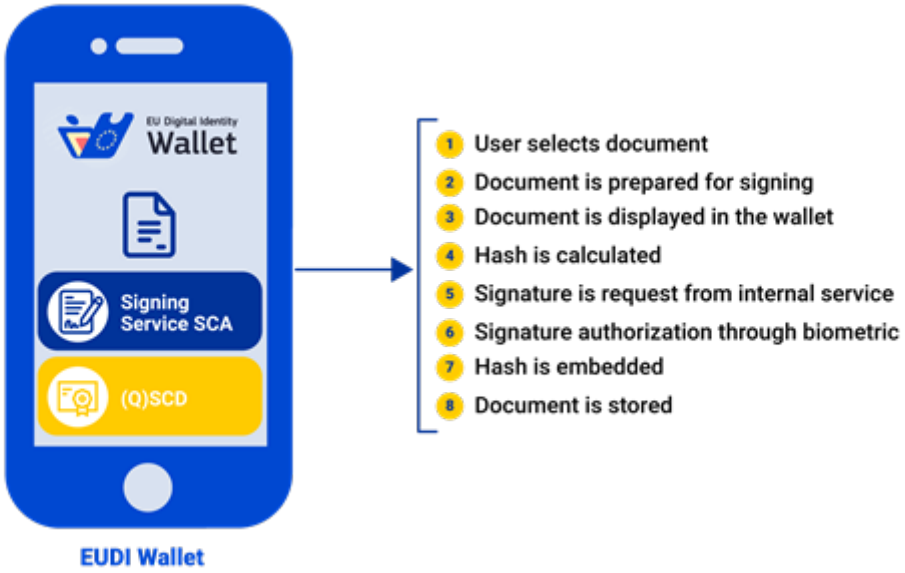
4.3.1.2 Remote Signing with Local SCA (Wallet as SCA)

The user initiates signing from the wallet, which also acts as the SCA. The document is presented to the user within the wallet for review and consent. After approval, the wallet forwards the signing request to a remote SCD that produces the signature and returns the result.



4.3.1.3 3) Local Signing

The user initiates signing from the wallet, which also acts as the SCA. The document is presented to the user within the wallet for review and consent. After approval, the signature is created locally using a SCD integrated in the user's device.



4.3.2 QTSP-centric Signing and Sealing Model

In the QTSP-centric model, the trust service provider operates the signing or sealing process. The cryptographic key material used for signature or seal creation is generated, stored, and used within infrastructure controlled by or on behalf of the QTSP, typically in secure hardware environments. From the perspective of the user, signing and sealing are therefore remote operations. External components interact with the trust service through defined interfaces, while the cryptographic operation itself is performed within the QTSP-controlled environment.

Within this architecture, the EUDI Wallet may act as a client-side orchestration component. It can authenticate the user, capture user intent, and trigger a signing operation. However, it does not operate the signature creation environment, does not manage signing credentials, and does not assume the responsibilities of a trust service provider.

In this model, the QTSP remains responsible for identity binding, credential issuance, and compliance with applicable ETSI standards. Signature or seal creation data remains under controlled conditions consistent with the required assurance level, and activation mechanisms enforce the conditions required for advanced or qualified signatures, including sole control where applicable.

The pilot implementation aims to remain technically aligned with qualified signing requirements. Pilot trust validation is described below and relies on consortium trusted lists.

4.3.3 CSC Interoperability Profile for Remote Signing and Sealing

For remote signing and sealing flows, WE BUILD uses the Cloud Signature Consortium (CSC) interoperability framework. CSC APIs expose standardised interfaces that allow wallets and client applications to interact with QTSP-operated signing services. The detailed WE BUILD CSC interoperability profile will be defined in a WBCS. That specification will describe the concrete integration details, including authorisation mechanisms, endpoints, supported formats and algorithms, and interoperability constraints used in the ITB. Until such a profile is published, CSC

API v2.2.0.0 serves as the base reference specification for CSC-based interactions.

During the pilot phase, trust validation relies on consortium reference trust mechanisms. Wallet components or external SCAs validate participating QTSPs and trust anchors using WE BUILD trusted lists. The reference trusted list may include participating QTSP entries and their registered issuing certificate authorities for pilot purposes.

When a signing request is processed, the SCA validates the signer's certificate chain against issuing certificate authorities listed in the WE BUILD trusted list and checks the QTSP status within the pilot trust framework. Revocation status is validated using OCSP responders or CRL distribution points operated by participating QTSPs. Where registration status checks are required, registrar processes are simulated through mock registrar services and endpoints.

4.3.4 Organisational Signing: Individuals Signing on Behalf of a Company

WE BUILD supports signing scenarios where an individual signs on behalf of an organisation. This model reuses the wallet-centric and QTSP-operated signing approaches described above. The wallet provides the user interface for document review and approval, while the QTSP performs the signature or seal creation within its controlled environment.

In these scenarios, the transaction must bind both the natural person and the organisation represented. The natural person identity is represented by the PID, while the organisation context is represented through the EBWOID, which acts as the cross-border minimum organisation identifier.

At signing time, identifiers or references to both the PID and the EBWOID are included in the transaction data presented to the user and subsequently authorised or signed. This ensures that the resulting signature or seal can be unambiguously linked to both the individual and the organisation.

The exact representation of these bindings is use-case specific and will be defined in rulebooks and WBCS (see Chapter 5 for the semantic and schema model).

4.4 Secure Communication Channel

This section describes how secure message exchange is integrated into the WE BUILD wallet ecosystem.

In WE BUILD, the secure communication channel is implemented through Qualified Electronic Registered Delivery Services (QERDS) operated by QTSPs. Whenever legal-grade delivery assurance is required, messages are routed through QERDS. QERDS providers ensure mutual authentication, end-to-end integrity and confidentiality, and interoperability across access points. This "registered delivery" pattern is positioned as an enabler for interactions between and across public sector bodies and economic operators.

Because the QERDS and the EU Digital Directory designated for the production European Business Wallet are not yet available, WE BUILD designates the pre-production QERDS specified by WP4 for use in WE BUILD business wallets. For reference, see the QERDS documentation.

4.4.1 From “Registered Delivery” to “Digital Identity Wallets”

As a baseline, a classic B2G/B2B situation is used: an authority notifies an economic operator, the economic operator responds, and the relying party requires evidence. With QERDS, both sides use their QERDS providers to register sending and receiving, so that delivery is not just transport, but is a process that produces trustworthy evidence.

In this model, WE BUILD takes the next step: wallets become the user-facing endpoints (“wallet-centric delivery”). The sender wallet and recipient wallet remain the places where users read, approve, and manage messages, or where they configure connections to backend systems to perform these actions. QERDS providers form the delivery layer underneath, handling routing, inter-provider exchange, and evidence creation, while wallets provide identity/authentication and user control.

4.4.2 Technical Flow (WE BUILD High-Level)

WE BUILD follows the QERDS architecture decomposition and the four-corner delivery pattern:

1. Sender identification and authentication is performed at the sender’s QTSP (wallet-driven).
2. Message submission is performed from the sender’s wallet or connected backend system to the sender QERDS (QTSP A).
3. Discovery of the recipient’s QERDS endpoint and capabilities is performed via common services (e.g., the WE BUILD Digital Directory, simulating the EU Digital Directory from the EBW proposal).
4. Handshake and relay is performed between QTSP A and QTSP B (QERDS-to-QERDS interoperability).
5. Recipient notification is issued, followed by recipient authentication at QTSP B.
6. Consignment and handover of the message and its metadata is performed to the recipient’s wallet or connected backend system.
7. Evidence is made available to sender and recipient wallets (submission/dispatch and receipt/consignment or non-delivery). Evidence is protected by qualified sealing and, where required, qualified timestamping. Where applicable, the evidence can be pushed to the sender’s and the recipient’s backend systems as well.

4.5 Enterprise and System-to-System Wallet Interactions

Some WE BUILD scenarios involve interactions between backend systems rather than direct end-user actions. In these cases, wallet functionality may be integrated into enterprise platforms, APIs, or automated services.

This is particularly relevant for EBW scenarios such as supply chain credentials, Digital Product Passports, and automated B2B or B2G data exchange. In such cases, credential issuance and presentation may be initiated by backend systems while still following the interoperability patterns defined in this blueprint.

Although the interaction is system-driven, the same trust framework, credential formats, and verification mechanisms apply as in user-driven wallet interactions.

5. Information Inside the Wallet

While the previous chapter describes how wallets interact with services, this chapter describes the data and semantic structures used inside the wallet and in the exchanged attestations.

5.1 Semantic Model of the European Business Wallet

The semantic model is organised into three layers:

1. Terminology — defines terms and their abstract concepts and establishes relationships between them.
2. Vocabulary — defines classes, properties, and individuals linked to the terminology.
3. Attestation mapping — maps vocabulary terms to the elements used in attestations.

The terminology and vocabulary layers are independent of attestation formats, while the mapping layer depends on the format used.

Currently, only W3C VCDM 2.0 supports machine-readable semantic mappings directly within credentials. For mDoc and SD-JWT-VC, the meaning of data fields must instead be defined in attestation rulebooks. In these cases, semantic interoperability between attestations is not automatically enforced.

5.1.1 WE BUILD Terminology

The WE BUILD terminology is published online and serves as a reference model for the terminology of the European Digital Identity Framework. The terminology is defined using the Simple Knowledge Organisation System (SKOS).

5.1.2 European Business Wallet Vocabulary

The European Wallet vocabulary is maintained in GitHub. It is defined using the Web Ontology Language (OWL), which specifies classes, properties and individuals of the vocabulary.

To support semantic interoperability, credential subjects used within the EBW framework are modelled in the vocabulary. These vocabulary terms are then mapped to the corresponding elements used in attestations.

If the credential format supports machine-readable semantic contexts, the mapping between credential data and the vocabulary can be embedded in the credential itself. Otherwise, the meaning of the data fields must be defined in attestation rulebooks, and the rulebook owner is responsible for mapping those fields to the vocabulary definitions.

Reuse of existing vocabularies The EBW vocabulary defines the domain-specific vocabulary used in the WE BUILD attestations. Existing vocabularies are reused where possible, including those for credential metadata, proof mechanisms, security, decentralised identifiers, and credential status.

Domain vocabularies from other sectors may also be reused (for example, digital product passports, supply chains, education, railway and data spaces).

5.2 Attestation Rulebooks and Credential Schemas

WE BUILD defines rulebooks and credential data schemas for the attestations used in the project's use cases. Rulebooks describe requirements, roles, processes, and conformance criteria for specific attestations. They also define how credential data fields relate to the semantic vocabulary used in the project.

Credential schemas define the structure of credential data and support implementers in producing interoperable credentials and validating that the data follows the agreed format.

Rulebook descriptions are currently provided by the use cases using a common template and are maintained in the project collaboration portal. As part of the ongoing work to formalise rulebooks and credential schemas, these descriptions will be consolidated in a shared repository such as the WE BUILD Attestation Rulebooks repository, including rulebooks for key credentials such as PID and EBWOID.

6. Trust, Security and Governance

The previous chapter described the structure of the information stored in wallets and exchanged as attestations. This chapter describes the trust infrastructure that allows ecosystem participants to validate those attestations.

6.1 Trust Ecosystem

The trust infrastructure for the EUDI and EBW ecosystem is based on three complementary processes: registration/onboarding of participants, notification of certain entities to the European Commission, and publication of Trusted Lists (or Lists of Trusted Entities) that provide cryptographic trust anchors for validation.

6.2 Establishing Trust Between Participants

WE BUILD defines the onboarding processes (how entities get registered), the trust framework (which rules apply), the PKI architecture (which certificates are used and how), the APIs used to query trust information programmatically, and the trust evaluation logic used by participants at runtime.

The infrastructure is based on the Trusted List model defined in the eIDAS Regulation and the ARF. It follows the European model in which a List of Trusted Lists (LoTL) points to Trusted Lists. Each Trusted List contains entries for authorised participants such as PID Providers, Attestation Providers (QEAA, PuB-EAA, non-qualified EAA), Wallet Providers, and Relying Parties.

The onboarding processes define how participants join the ecosystem. This includes how:

- **Relying Parties** register, accept policies and configure access controls

- **PID and Attestation Providers** register, declare supported attestation types and obtain registration and access certificates
- **Wallet Providers** register and issue wallet instance attestations
- **Trust Service Providers** register and publish relevant certificates

Once onboarding is completed, participants use the trust infrastructure to evaluate each other during normal operation. WE BUILD therefore defines a set of trust evaluation scenarios covering how participants verify each other at runtime.

These scenarios include:

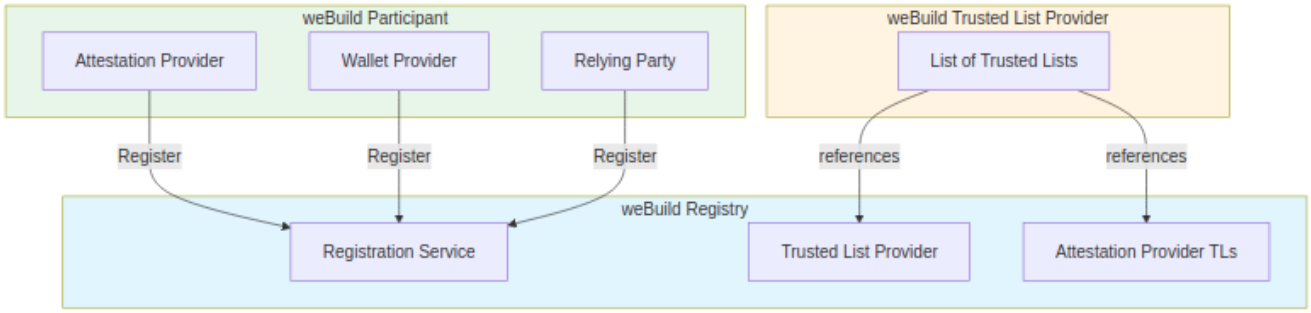
- a Wallet Unit evaluating a Credential Issuer before requesting a PID or attestation
- a Credential Issuer evaluating the Wallet Unit before issuing
- a Wallet Unit evaluating a Relying Party before presenting attributes
- a Relying Party evaluating presented credentials (PID, QEAA, PuB-EAA, non-qualified EAA)
- discovery and consumption of the LoTL and TLs.

For detailed information on authorities, registries and responsibilities, see Appendix C - Trust Ecosystem.

For reference on relying party access certificates and relying party registration certificates, see the RPAC/RPRC documentation.

6.2.1 Trust infrastructure architecture (overview)

In the Appendix - Trust Ecosystem there is a diagram that summarises the roles of Member State and European Commission, the split between registration and notification, and how Trusted Lists and the LoTL are produced and consumed. A simplified version used in WE BUILD is shown below.



WE BUILD participants select the registry in which they register.

6.3 Revocation

Revocation ensures that attestations that are no longer valid can no longer be trusted or used.

WE BUILD distinguishes between attestation revocation, which is handled by issuers, and revocation or withdrawal of providers and services, which is reflected in the trust infrastructure.

6.3.1 Technical realisation

Revocation of PID, EBWOID and attestations is implemented by issuers. In WE BUILD, attestation revocation follows the agreed mechanism defined in the ADR on Attestation Revocation, based on the IETF Token Status List and aligned with OpenID4VC HAIP.

Short-lived attestations (valid for 24 hours or less) are not subject to revocation.

Revocation or withdrawal of providers and services is reflected in the trust infrastructure through status changes in Trusted Lists and, where applicable, invalidation of certificates.

6.3.2 Provider Obligations

To maintain a trusted ecosystem, PID and EBWOID providers agree to:

- Define and publish revocation policies.
- Ensure that only the issuing authority can revoke its attestations.
- Publish revocation status information within a reasonable time frame.

6.3.3 Conditions for Mandatory Revocation

According to the rules, a provider must revoke without delay if:

- The holder explicitly requests it.
- The security of the wallet app itself (the unit certificate) is compromised.
- Any of the specific situations defined in the provider's public policy occur.

7. Architecture Governance: ADRs and WBCS

While the previous chapter described the operational trust infrastructure, this chapter describes the governance model used to define and maintain the technical architecture of the WE BUILD ecosystem. In a project as large as WE BUILD, interoperability between independently developed components must be ensured without requiring every developer to participate in all coordination meetings.

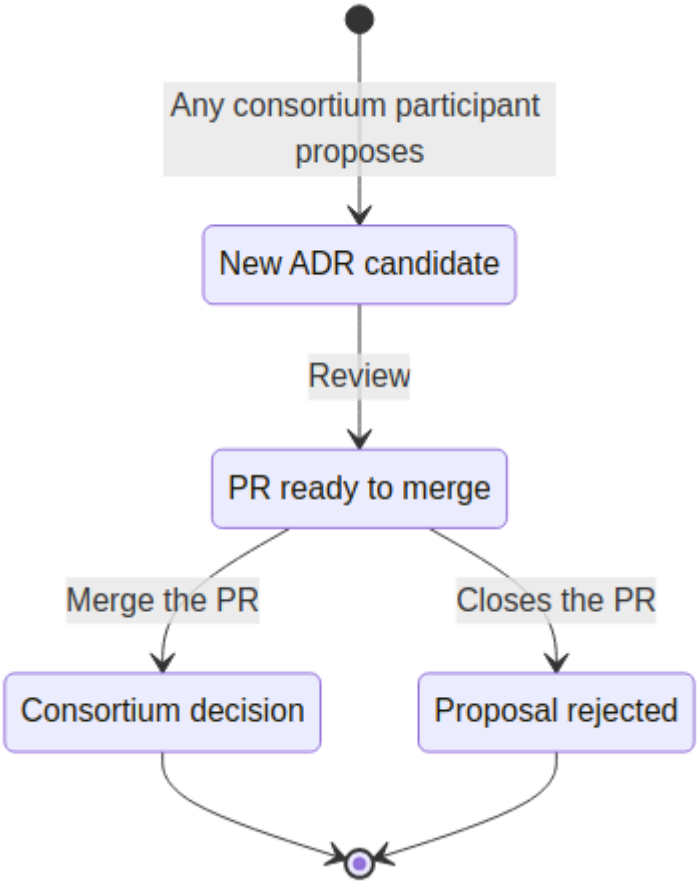
To maintain alignment, the project uses a technical governance model based on consensus, commitment and clear documentation. Technical choices are driven by the needs of the 13 use cases implemented in the project.

7.1 Architectural Decision Records (ADR)

The ADRs is essentially our project's "logbook" for major decisions.

- Purpose: The ADR process is where we formally capture and justify significant technical choices, such as which specific protocols and formats to use. Instead of having these decisions buried in a slide deck or a long email chain, we document the rationale and context so that everyone can understand the "Why" behind a choice.

- Classification: We maintain a lightweight ADR for any software-related decision that affects how different systems work together (interoperability). This ensures alignment with external rules like the eIDAS Regulation and the ARF.
- Lifecycle: ADRs are managed on GitHub (webuild-consortium/wp4-architecture). They move from a "Proposed" state to "Accepted" once the Architecture Group and relevant stakeholders reach consensus.

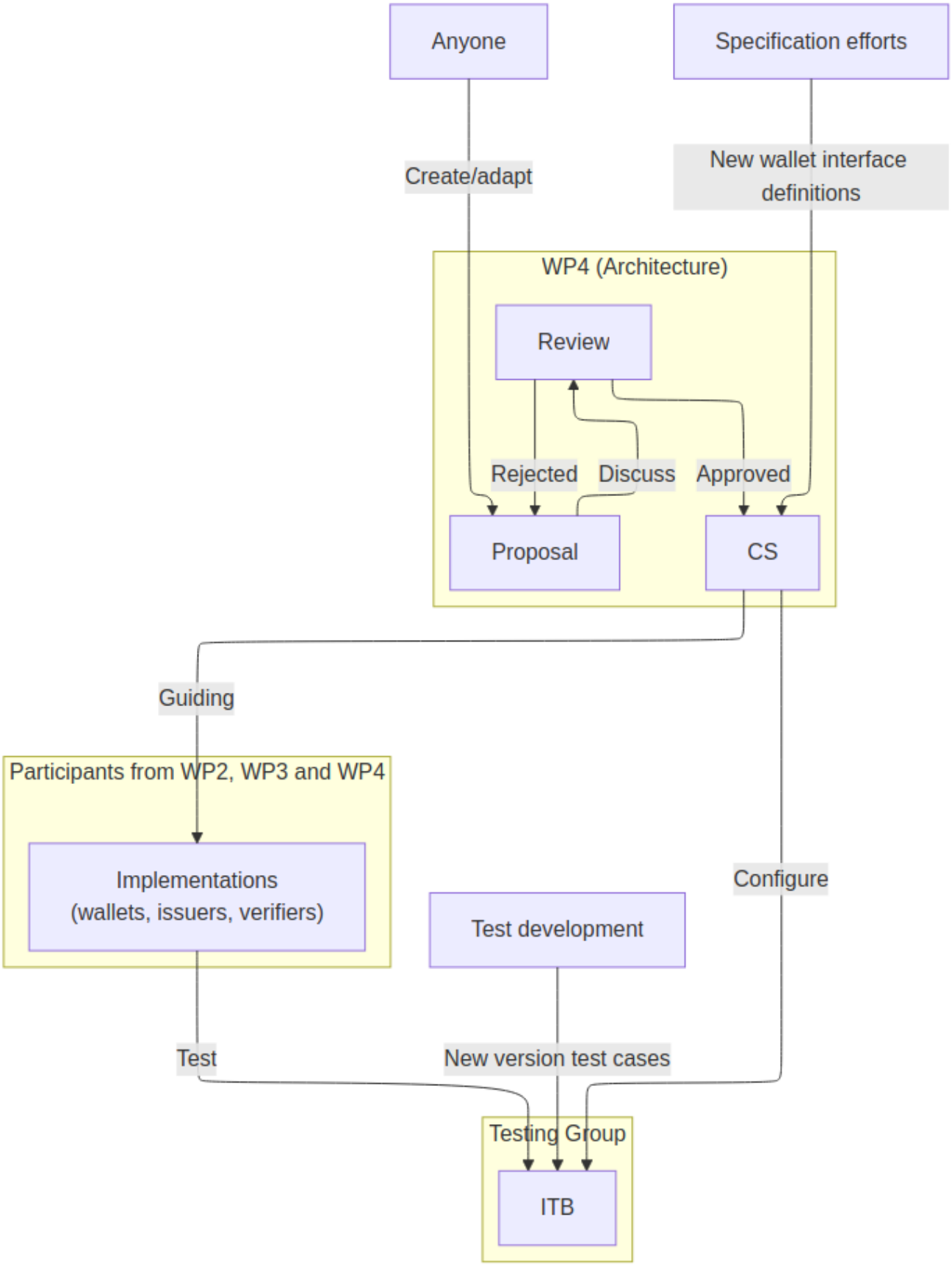


7.2 WE BUILD Conformance Specifications (WBCS)

If ADRs capture the rationale ("why"), the WBCS define the implementation requirements ("how").

- Operationalising Intent: We use the WBCS to turn high-level architectural goals into detailed technical rules. These specifications define the exact interfaces for wallets, issuers, and verifiers.
- A Commitment to Implement: This is the most important part: An approved WBCS is **not just a suggestion**. When a specification is approved, it signifies a commitment from the participating organizations to actually build that interface into their services.
- Defining Implementation Requirements: Because the WBCS define how interfaces and protocols must be implemented, they allow us to achieve interoperability across the whole consortium. If you follow the WBCS, you avoid building an "interoperable island" where your service only works with a few specific partners.
- The Link to Testing: Our Interoperability Testbed (ITB) uses these specifications as its primary rulebook. Implementations that do not follow the WBCS will not pass the ITB tests and are

therefore not eligible for pilot participation.



7.3 Document Lifecycle

WE BUILD moves fast, and our documentation needs to keep up. We don’t wait for "perfect" documents; we iterate as the use cases mature.

- The Blueprint as a Living Framework: This Blueprint (D4.1) sets the high-level structure, but it is supported by the more agile ADRs and WBCS that live on GitHub. As we learn, we update these records and specifications.
- The Hybrid Working Flow: To keep things moving, we use a "hybrid" approach to our document lifecycle:
  - GitHub: This is our source of truth for all accepted specifications and decision records.
  - Slack: The ITB uses a dedicated channel for implementation support, where developers can ask questions and help each other in real-time.
  - Meetings: We hold interface-alignment meetings to discuss progress, resolve gaps, and gain final agreement on new specifications.
- Maturing Together: As the project moves forward, we will add more detailed definitions to the documentation stack. This approach allows the Blueprint to evolve from a high-level architectural reference into a practical guide for implementing the WE BUILD ecosystem.

8. Testing and the Interoperability Testbed (ITB)

Once architectural decisions and specifications are defined, interoperability must be verified in practice. This chapter describes how interoperability is verified through the WE BUILD testing strategy and the Interoperability Testbed (ITB).

8.1 Testing Strategy

The Architecture Group coordinates the architectural building blocks and ensures alignment with the project use cases.

The Testing Group develops test cases and test suites for:

- Generic test cases based on WBCS.
- Functional test cases for required features (based on WBCS and, when needed, rulebooks and/or data schemas).
- End-to-end and piloting test cases for WP2/WP3 use cases (based on existing WBCS, rulebooks and data schemas).

To implement tests in the ITB, the Testing Group needs the specification artefacts: WBCS, rulebooks, data schemas, namespaces, and related metadata. The Architecture Group ensures that these artefacts are complete and consistent with the overall architecture and supports WP4 groups and WP2/WP3 use cases in providing the required input.

Most specification artefacts are produced within WP4:

- The Semantics Group: attestations (data schemas, namespaces, and relevant rulebook parts).
- The Wallet, PID/EBWOID and QTSP Group: WBCS and commitment to implement them.
- The Architecture Group: Architecture Decision Records (ADRs) that define the allowed scope for WBCS.
- The Trust Infrastructure Group: validation and verification requirements to be reflected in test cases.

For piloting-specific test suites, the Testing Group collaborates directly with the relevant use case(s). The Architecture Group acts as a facilitator to ensure consistency across the involved specifications.

8.2 Test Requirements

Test cases are derived from the WBCS.

WBCS must stay within the scope defined by the published ADRs. If a WBCS needs functionality beyond that scope, it requires an ADR discussion. Testing focuses on features implemented by multiple parties, since interoperability requires multi-party implementations.

Implementing participants discuss WBCS together with the use cases that require the functionality.

The ITB initially includes two credential-agnostic test suites:

- Issuing (based on OpenID4VCI v1.0)
- Verifying (based on OpenID4VP v1.0)

If a use case requires different functionality, it can propose a new or adapted (draft) WBCS. Once the WBCS and required supporting artefacts are available, the Testing Group implements the corresponding test cases in the ITB.

Some test cases require additional artefacts beyond the WBCS, such as rulebooks for attestation-specific requirements, and the corresponding data schemas, namespaces, and metadata.

When the required artefacts are available, the Testing Group implements the test cases in the ITB and communicates their availability to the consortium.

8.3 Additional Documentation

The ITB on GitHub

A user guide on how to onboard and execute tests.

Documentation on the ITB and integrations

9. What's Next & Scaling Up

This document, together with the initial attestation data models and schemas, establishes a common technical baseline for the wallet ecosystem. The architecture will continue to evolve throughout the project as the focus shifts towards implementation and testing.

WP4 - General Capabilities operates on a defined timeline with key milestones and deliverables while adapting to feedback from the use cases as well as business, regulatory and technological developments. The deliverable is expected to evolve through updates and refinements as new ADRs and WBCS are added or revised. The document therefore serves as a living reference for the WE BUILD architecture.

The roadmap for months 8—19 focuses on maturing the trust infrastructure and validating cross-border interoperability through the project's automated testbed.

Key Milestones and Deliverables (Months 8—19)

Month	Type	Reference and Title	Connection to D4.1
10	Deliverable	D4.3 Attestation Data Models & PID/EBWOID Rulebook	Finalizes the data structures for the issuance patterns defined in D4.1
11	Milestone	MS5 WP2 Pilot Design Complete (Business)	Finalizes the scope of business use journeys based on D4.1 architectural patterns
11	Milestone	MS10 WP3 Pilot Design Complete (Payments)	Finalizes the scope of payments and banking use journeys based on D4.1 architectural patterns
12	Deliverable	D2.1 WP2 Pilot Design Report & PID/EBWOID issuance rulebook	Maps user journeys to the D4.1 reference implementations
12	Deliverable	D3.1 WP3 Pilot Design Report	Maps payment journeys to the D4.1 reference implementations
13	Deliverable	D4.4 Trust Infrastructure Guidelines	Details the technical signature/seal flows introduced in D4.1
17	Milestone	MS6 WP2 Pilot Implementation Complete	Validates local business infrastructure against D4.1 specifications
17	Milestone	MS11 WP3 Pilot Implementation Complete	Validates local payment infrastructure against D4.1 specifications
19	Milestone	MS7 WP2 Cross-Border Readiness	Proves interoperability of WP2 and WP4 business ecosystem
19	Milestone	MS12 WP3 Cross-Border Readiness	Proves interoperability of WP3 and WP4 payments and banking ecosystem

Month	Type	Reference and Title	Connection to D4.1
19	Milestone	MS17 Wallets & Trust Infrastructure Ready	Final evidence of wallet/trust interoperability as per D4.1

Alongside these milestones and deliverables, WP4 will continue to iterate on the architecture, specifications and supporting artefacts throughout the project. All WP4 groups will incorporate feedback from the piloting phase and adapt to new requirements and emerging EUDI standards and other technological developments. The ITB will remain a vital tool for continuous integration and testing, ensuring that the solutions remain interoperable, secure, and scalable.

Appendix A. Glossary

Terms and Definitions

This appendix defines the key terms, regulatory frameworks, and technical specifications utilised throughout the WE BUILD ecosystem.

While this document avoids abbreviations as much as possible, commonly used abbreviations are included for reference.

Term	Abbreviation	Definition
Architectural Decision Record	ADR	A document used to capture and justify significant technical choices. ADRs serve as the project's "logbook" to ensure transparency regarding the rationale behind protocol and standard adoption.
Architecture and Reference Framework	ARF	The reference architecture for the European Digital Identity Wallet ecosystem published by the European Commission in cooperation with the Member States. It defines roles, trust models, protocols and interoperability requirements for the ecosystem.
Attestation Rulebook	-	A document describing the governance, requirements and semantic interpretation of a specific attestation type, including how credential data maps to vocabulary terms and schemas.

Term	Abbreviation	Definition
Blueprint	—	The high-level architecture and integration document (D4.1) describing the WE BUILD ecosystem, architectural patterns, interaction flows and governance model.
Business Wallet Unit Attestation	BWUA	A specific type of Wallet Unit Attestation issued for a European Business Wallet (EBW) instance.
EAA Provider	—	An entity that relies on authentic sources of information to issue attestations to a wallet.
EBW Instance	—	A unique deployment or installation of a European Business Wallet (EBW) solution, controlled by an Owner (legal person or economic operator).
EBW Provider	—	A Wallet Provider specifically authorized to issue and manage European Business Wallets (EBW).
EBWOID Provider	—	An entity responsible for verifying the identity of a legal person or economic operator and issuing EBW Owner Identification Data (EBWOID).
EBW Owner Identification Data	EBWOID	A set of attributes used to uniquely identify a legal person or economic operator within the European Business Wallet ecosystem.
Economic operator	—	Any natural or legal person or public entity which offers products or services on the market; the primary user of the European Business Wallet.

Term	Abbreviation	Definition
Electronic Attestation of Attributes	EAA / QEAA / PuB-EAA	Digital credentials that prove specific attributes (e.g., professional qualifications, representation rights) with either qualified (QEAA) or public sector body-issued (PuB-EAA) or non-qualified (EAA) legal status.
Electronic Identification, Authentication and Trust Services	eIDAS / eIDAS 2.0	The legal framework for electronic identification and trust services for electronic transactions in the European Single Market.
European Business Wallet	EBW	A wallet designed for economic operators or public sector bodies to manage business data such as mandates, electronic invoices, and administrative and professional documents and notifications.
European Digital Identity Wallet	EUDI Wallet	A mobile or cloud-based solution for natural persons to manage and share identity data.
EUDIW Instance	—	A specific deployment of an EUDI Wallet solution for a natural person.
~~**Holder**~~	—	See *Wallet User* instead.
Interoperability	-	The ability of independently developed systems and components to exchange information and correctly interpret the exchanged data.
Interoperability Testbed	ITB	The automated testing environment used in WE BUILD to verify that implementations conform to the agreed specifications and remain interoperable.
~~**Issuer**~~	—	See *EAA Provider* instead.

Term	Abbreviation	Definition
Large Scale Pilot	LSP	A project funded by the European Commission to test the practical implementation of the EUDI Wallet framework across various cross-border use cases.
Legal Person	—	An entity (such as a corporation or public body) recognized by law as having rights and duties, distinguished from a natural person.
~~**Legal Person Identification Data**~~	LPID	See **EBW Owner Identification Data** instead.
Level of Assurance	LoA	A classification of the degree of confidence in the electronic identification of a natural person, a legal person, or a natural person representing a legal person. Recognised levels are: Low, Substantial, High.
List of Trusted Lists	LoTL	A list that references national or ecosystem Trusted Lists, allowing participants to discover and validate trusted entities.
Natural Person	—	An individual human being acting in their own capacity.
Owner	—	The legal person or economic operator that has legal control over and responsibility for an EBW Instance.
Personal Identification Data	PID	A mandatory set of attributes issued to a natural person to uniquely identify them at Level of Assurance (LoA) High.
PID Provider	—	An entity responsible for verifying the identity of a natural person and issuing Personal Identification Data (PID).

Term	Abbreviation	Definition
Qualified Electronic Registered Delivery Service	QERDS	A secure communication channel that provides legal evidence of the handling of transmitted data.
Qualified Trust Service Provider	QTSP	A regulated entity providing electronic trust services (e.g., signatures, seals, or delivery services) with full legal effect under eIDAS.
Relying Party	RP	An entity that requests and receives attestations from a wallet to verify specific attributes or identities.
Selective Disclosure JSON Web Token	SD-JWT	A format allowing holders to share only specific parts of a credential while keeping other data private.
Trust Framework	—	The set of governance rules, standards, and trust infrastructure used to establish and verify trust relationships between ecosystem participants.
Trusted List	TL	A machine-readable list of trusted service providers or entities used to validate trust relationships within the ecosystem.
~~**Verifier**~~	—	See *Relying Party* instead.
Wallet Application	—	The user-facing software component of a Wallet Solution providing the interface for managing credentials.
Wallet Core Component(s)	—	The technical module(s) of a Wallet Solution handling cryptographic operations and protocol implementations.
Wallet Instance	—	A specific operational instance of a wallet solution running on a device or cloud environment.

Term	Abbreviation	Definition
Wallet Instance Attestation	WIA	A short-lived, signed information object issued by a Wallet Provider that contains information about the Wallet Instance. It is device-bound and presented to PID or Attestation Providers to authenticate the instance, but it does not require a WSCD/WSCA for key management and does not contain revocation information.
Wallet Provider	—	An organization that provides a Wallet Solution and manages its lifecycle.
Wallet Secure Cryptographic Device / Application	WSCD / WSCA	The hardware or software environment used to manage cryptographic keys securely within the wallet.
Wallet Solution	—	A specific implementation of a wallet consisting of a Wallet Application and Wallet Core Component(s).
Wallet Unit Attestation	WUA	A signed information object issued by a Wallet Provider that describes the capabilities and security properties of a Wallet Unit (especially the WSCD/WSCA). It is device-bound and allows PID or Attestation Providers to verify compliance, bind credentials to the unit, and check for revocation.
Wallet User	—	The natural or legal person that controls and operates a wallet instance.
WE BUILD	—	The consortium and project focused on pioneering the European Business Wallet and EUDI Wallet use cases.

Term	Abbreviation	Definition
WE BUILD Conformance Specifications	WBCS	Detailed technical rules that operationalize architectural intent. Approval of a WBCS signifies a commitment from partners to implement those interfaces.

Appendix B. Document History

Changes

Version	Date	Author	Description
1.0	2026-03-20	**Authors** Sarah Amandusson, Digg, SE Sander Dijkhuis, Cleverbase, NL George Fourtounis, GRNet, GR Benjamin Hansson, iGrant.io, SE Leif Johansson, SIROS Foundation, SE Svilena Rakshieva, Evrotrust Technologies, BG Sebastian Elfors, IDNow, FR George J Padayatti, iGrant.io, SE Giuseppe De Marco, Dipartimento per la trasformazione digitale, IT Ronald Koenig, Spherity, DE **Contributing parties** Steffen Piel, Governikus, DE Malin Norlander, Bolagsverket, SE Lal Chandran, iGrant.io, SE Aleksandar Simsic, ICTU, NL Esther Maakay, Signicat, NL Hristian Daskalov, Evrotrust Technologies, BG Thodoris Papadopoulos, GRNet, GR Eelco Klaver, Credenco, NL Viky Manaila, Intesi Group, IT Andreas Abraham, Validated ID, ES Alejandro Nieto, Digitel TS, ES Stefan Hadjistoytchev, Evrotrust Technologies, BG Andrew Freund, D-Trust, DE Miguel Aguilar, Bolagsverket, SE Dr. Oliver Froitzheim, Bundesanzeiger Verlag GmbH, DE Sarah Gräfer, Bundesanzeiger Verlag GmbH, DE **Reviewers** Andriana Prentza, University of Piraeus, GR	First complete version with full content

Version	Date	Author	Description
0.1	2025-12-08	Sarah Amandusson	Initial structure

Appendix C. Trust Ecosystem

The trust infrastructure for the EU Digital Identity and European Business Wallet ecosystem rests on three distinct but complementary processes: **registration/onboarding** of participants, **notification** of certain entities to the European Commission, and **publication of Trusted Lists** (or Lists of Trusted Entities) that provide cryptographic trust anchors for validation. WE BUILD aligns with the EUDI Wallet Architecture and Reference Framework (ARF) and the trust-infrastructure model described in the WP4 Trust Group deliverables.

Trust infrastructure authorities and registries

- **Member State Registrar:** Manages registration and operational authorization of **PID Providers**, **Attestation Providers**, and **Relying Parties**. Registration yields registry entries (used for entitlement verification and online lookup via common APIs such as TS5/TS6) and triggers access certificate issuance.
- **European Commission:** Compiles, signs/seals, and publishes Trusted Lists for Wallet Providers, PID Providers, Access Certificate Authorities (Access CAs), and Providers of Registration Certificates. It maintains the **List of Trusted Lists (LoTL)** and publishes LoTL location and trust anchors in the Official Journal of the European Union (OJEU).
- **Member State Trusted List Provider (MS TLP):** Compiles, signs, and publishes national Trusted Lists for **non-qualified EAA Providers** and **Member State QTSP Trusted Lists** for **QEAA Providers** (per Article 22 eIDAS), and submits the Trusted List URLs to the Commission for inclusion in the LoTL.
- **Access Certificate Authority:** Issues access certificates to registered entities (PID Providers, Attestation Providers, Relying Parties). Notified by Member States to the Commission; does not register with Registrars.
- **Provider of Registration Certificates:** Optionally issues registration certificates that detail entitlements; notified by Member States to the Commission.

Registration vs Trusted List publication: Registration defines *who is allowed to do what* (entitlements, attributes, intended use) and is consumed via registries and optional registration certificates. Trusted List publication establishes *cryptographic trust anchors* (keys, certificates) and, via profile-specific extensions defined in **ETSI TS 119 602** (for example the Pub-EAA and national non-qualified EAA Provider LoTE profile in Annex H, including its additionalInfo structures), can also publish **which attestation types an Attestation Provider is authorised to issue**. Trusted Lists follow ETSI TS 119 612 and TS 119 602 (Lists of Trusted Entities) and are consumed per ETSI TS 119 615. Wallet Providers, Access CAs, and Providers of Registration Certificates are **not** registered with Registrars; these entities are notified by Member States to the Commission.

Responsibilities matrix

The Task 2 trust-infrastructure schema defines the following responsibilities matrix for registration

and Trusted List compilation:

Entity Type	Registration Process	Trusted List Compilation (EC / MS TLP)	Member State TLP Role
PID Provider	Register with MS Registrar	European Commission (EU-level TL for PID Providers)	None (no national TL for PID Providers)
Attestation Provider	Register with MS Registrar	Member State / MS TLP (national QTSP TL for QEAA Providers; national TL for non-qualified EAA Providers)	Compiles, signs, and publishes national Trusted Lists (QTSP TL for QEAA Providers per Article 22; EAA Provider TL for non-qualified EAA Providers)
Relying Party (RP)	Register with MS Registrar	N/A (uses Access Certificates and Registry)	None (not listed in Trusted Lists)
Wallet Provider	Notification only (by MS to EC)	European Commission (EU-level TL for Wallet Providers)	Not applicable in MVP (notification from MS to EC only)
Access CA	Notification only (by MS to EC)	European Commission (EU-level TL for Access CAs)	Not applicable in MVP (notification from MS to EC only)
Reg. Cert. Provider	Notification only (by MS to EC)	European Commission (EU-level TL for Reg. Cert. Providers)	Not applicable in MVP (notification from MS to EC only)

This blueprint section mirrors the Task 2 responsibilities matrix so that architectural roles are consistent across WP4.

Working group scope: [MVP] and [MVP+]

In line with the EUDI Wallet ARF, the WP4 Trust Group focuses on defining **architectural patterns and profiles**, not on specifying Member State-specific policies or operating production infrastructure. To make this concrete, the trust and security work is scoped in two steps:

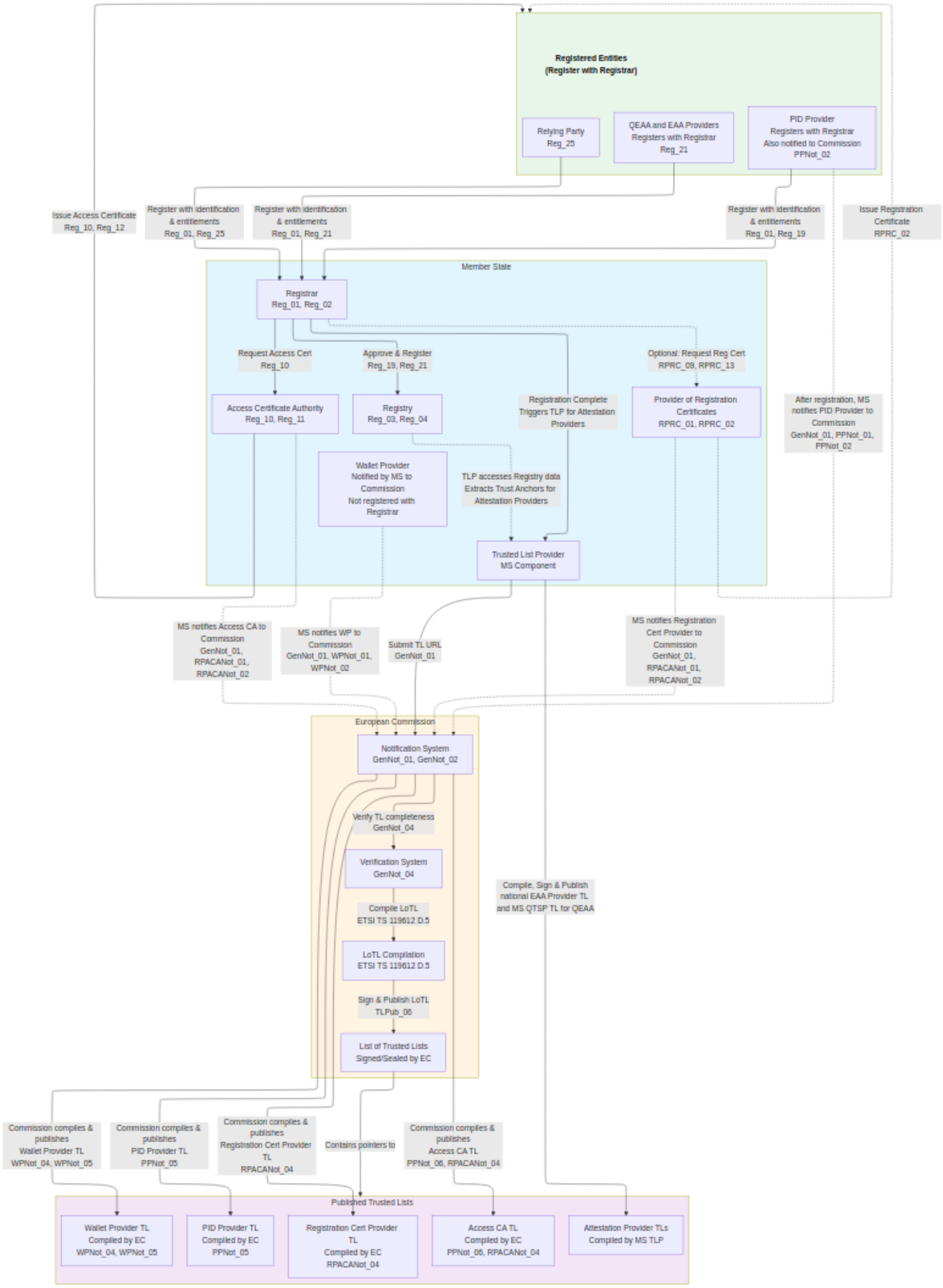
- **[MVP] (Minimum Viable Prototype)**:
  - Implements the **core onboarding scenarios** from Task 1 (Subtask 1.1) for PID Providers, Attestation Providers, Wallet Providers, Relying Parties, and Certificate Authorities.
  - Implements the **basic trust-registry scenarios** from Task 1 (Subtask 1.2) needed to create, publish and consume Trusted Lists / Lists of Trusted Entities and registry entries for these actors.

    - Demonstrates end-to-end flows for **registration, access-certificate issuance, trust-anchor publication and consumption**, reusing the ARF and ETSI TS 119 602/119 612/119 615 patterns without introducing new normative profiles.
- **[MVP+] (Extended prototype)**:
    - Completes the remaining Task 1 onboarding and trust-registry scenarios, including more advanced **evaluation, maintenance, revocation and discovery** cases.
    - Covers **richer combinations of participants and roles** (e.g. multiple types of Attestation Providers and more complex Relying Party ecosystems) while staying within the Task 2 trust framework and trust-infrastructure schema.
    - May introduce **pilot-specific configurations or conventions** (e.g. additional metadata, policy examples, or Trusted List extensions) as long as these remain compatible with the underlying ARF and ETSI models and are clearly marked as non-normative.

The **boundary of the working group** is therefore to: (a) define the trust-infrastructure architecture, profiles, and flows needed for [MVP] and [MVP+]; (b) document how to apply ETSI TS 119 602/119 612/119 615 and the ARF in these scenarios; and (c) leave Member State policy choices (approval criteria, national extensions, operational SLAs) and long-term production operation out of scope.

Trust infrastructure architecture (overview)

The following diagram summarises the roles of Member State and European Commission, the split between registration and notification, and how Trusted Lists and the LoTL are produced and consumed. Source: WP4 Trust Group Task 2, trust-infrastructure schema.



NOTE

WP4 is going to expose trust infrastructure depicted on this diagram, mimicking the infrastructure of at least one Member State. In the case of mimicking more than a single Member State, WE BUILD participants willing to register are going to be able to select a registry in which they are going to be registered, or the WP4 registrar, if

> single, is going to place them in one of the registries. For the sake of simplicity, in such a case, not all the technical components depicted in the diagram within a Member State will have to be multiplied; e.g. there may be multiple registries but a single trusted list across the countries to which all wallet-relying parties are provisioned.

Security Measures

From an architectural perspective, security in the wallet ecosystem is structured in four dimensions. (1) **Trust anchor layer**: cryptographic validation and key lifecycle management, including revocation of keys, certificates, and services. Trust Anchors are published by Trusted Lists and related mechanisms (ETSI TS 119 602/119 612/119 615), applying to the entities and roles described in the Trust Ecosystem (PID Providers, Attestation Providers, Wallet Providers, Access CAs, etc.). (2) **Identity assurance**: the level of assurance (LoA) of the identities involved is maintained across the full lifecycle of those identities. (3) **Device and execution environment**: the security of the devices and execution environments that host Wallet Instances and cryptographic material (WSCA/WSCD) is addressed by the wallet secure cryptographic application/device (WSCA/WSCD) architecture for wallet-side components, to be specified by the Architecture and Wallets groups (and, for remote WSCA/WSCD, together with the QTSP group). Secure environments operated by PID Providers, Pub-EAA and QEAA Providers, and Trust Service Providers (TSPs) are addressed by applicable eIDAS and ETSI requirements for TSP and provider infrastructure. (4) **Protocol and policy layer**: authentication (verifying who or what is acting) and authorization (what the subject is allowed to do, driven by policies) are realised in conformance with the ARF and related technical specifications, per application-specific flow and per attestation data format.

Authentication

Authentication in the WE BUILD architecture closely follows the standards and flows of the underlying protocols (**OpenID for Verifiable Credentials**, **ISO 18013-5**, and other application-specific communication protocols).

For **organizational entities** that register with the Registrar (PID Providers, Attestation Providers, Relying Parties), authentication is primarily established using **access certificates** issued by the Access Certificate Authority, validated against up-to-date Trusted Lists or Lists of Trusted Entities (ARF and ETSI TS 119 602 / 119 612 / 119 615). **Wallet Providers** are notified by Member States to the European Commission (they do not register with the Registrar); their authenticity is established via the Wallet Provider Trusted List and related attestations (e.g. Wallet Unit Attestation). Mutual trust is strictly required in application-specific protocol flow specifications, and consolidated through OpenID HAIP and ARF HLRs, with certificate-bound tokens and protocol-level message signatures along with endpoint authentication and message integrity.

According to the ARF, the Relying Party **cannot request the Wallet Unit Attestation (WUA) during the presentation flow**. Presentation requests address **PID and attestations** only (ARF Topic 1, **OIA_01**); the WUA is presented to the PID Provider or Attestation Provider **during issuance** of a PID or device-bound attestation, not to the Relying Party (ARF Topic 9, **WUA_03**, **WUA_05**, **WUA_05a**). The ARF explicitly states that there is no separate mechanism for the Relying Party to verify the revocation status of a Wallet Unit directly with the Wallet Provider (ARF Section 6.6.3.12). Trust in the Wallet Unit is therefore **mediated by a trusted third party**: the PID Provider or

Attestation Provider that belongs to a Trust Anchor (Trusted List) and that received the WUA at issuance. That trust is **indirect** from the Relying Party's perspective. The PID Provider or EAA Provider **periodically checks the revocation status** of the Wallet Unit to which it has issued credentials (using the revocation information in the WUA received at issuance; ARF Topic 9 **WUA_02**, ARF Section 6.6.2.4). If the Wallet Unit is revoked, the PID Provider or Attestation Provider **SHALL revoke** the credentials it issued to that Wallet Unit (Article 5, 4.(b), European Digital Identity Regulation; ARF Section 6.6.2.4). By verifying the revocation status of the PID or attestation, the Relying Party implicitly relies on the issuer's verification of the Wallet Unit.

For **attestation holders** (individual end-users and the corresponding wallets), authentication requirements leverage possession-based proofs and, where applicable, **identity schemes notified to the European Commission** that attest **Level of Assurance High** according to the eIDAS Framework. Local user authentication (e.g., via PIN, password, device biometrics) must fulfill the minimum Level of Assurance required by the credential or attestation type and is enforced by policy prior to any issuance or presentation flow.

Wallet Units are expected to combine protocol-specific authentication mechanisms (as per OpenID4VC and ISO 18013-5) with validation of trust anchors and up-to-date Trusted Lists, covering both participant and credential authenticity.

Authorization and policies

Authorization determines what actions a subject is permitted to perform after authentication. In line with the Task 2 trust framework and the EUDI Wallet ARF, WE BUILD **assumes** that **the default is "allow all"** at the ecosystem level, and that **policies (expressed via Trusted List extensions, registration/entitlement data, and optional registration certificates) can tighten this to an effective "deny all except explicitly allowed"** model for specific contexts and participants. When such policies apply, only the actions and attribute uses listed in the applicable allow-lists are permitted; all others are denied. Trust marks (for Credential Issuers, Wallet Solutions, and Relying Parties), together with Trusted List extensions and registration certificates, carry authorization semantics (e.g. authorised credential types, attribute groups, purposes, scope restrictions) and are used in policy evaluation and collision prevention as specified in the Task 2 trust framework and the trust-infrastructure schema.

According to the EUDI Wallet ARF v2.8, when present and applicable, policies and default authorisations may be overridden by **user will**: the Wallet Unit SHALL ensure the User approved the presentation of any attribute(s) prior to presenting those attributes and SHALL always allow the User to refuse presenting an attribute requested by the Relying Party or Verifier Wallet Unit (ARF Topic 6, **RPA_07**); if Relying Party authentication fails, the Wallet Unit SHALL either not present the requested attributes or give the User the choice to present or not (**RPA_06a**).

Certificates and cryptographic anchors

- **Trusted Lists / Lists of Trusted Entities (LoTE)** (ETSI TS 119 612, TS 119 602) are pivotal trust anchors in the ecosystem. LoTE entries publish the keys and related metadata for the entity types described in the Trust Ecosystem (Wallet Providers, PID Providers, Attestation Providers, Access CAs, Registration Cert Providers). Validation of trust service outputs against these lists SHALL follow **ETSI TS 119 615** (procedures for using and interpreting EUMS national trusted lists).

- **Access certificates** are issued by the Access Certificate Authority to registered PID Providers, Attestation Providers, and Relying Parties. Issuance SHALL comply with **ETSI TS 119 411-8**; the Authority SHALL comply with at least **ETSI EN 319 411-1** Normalised Certificate Policy (NCP) requirements. Each Relying Party receives a **separate access certificate per Relying Party Instance**. Access certificates authenticate entities in protocol exchanges and are validated by Wallet Units using the trust anchors in the Access CA LoTE entries.
- **Registration certificates** (optional) may be issued by the Provider of Registration Certificates to detail registration status and entitlements. When the User opts to verify RP (or issuer) registration, Wallet Units use the registration certificate when provided and/or registry lookup, as specified in ARF RPRC_16 to RPRC_21.

For reference on relying party access certificates and relying party registration certificates, see the RPAC/RPRC documentation.

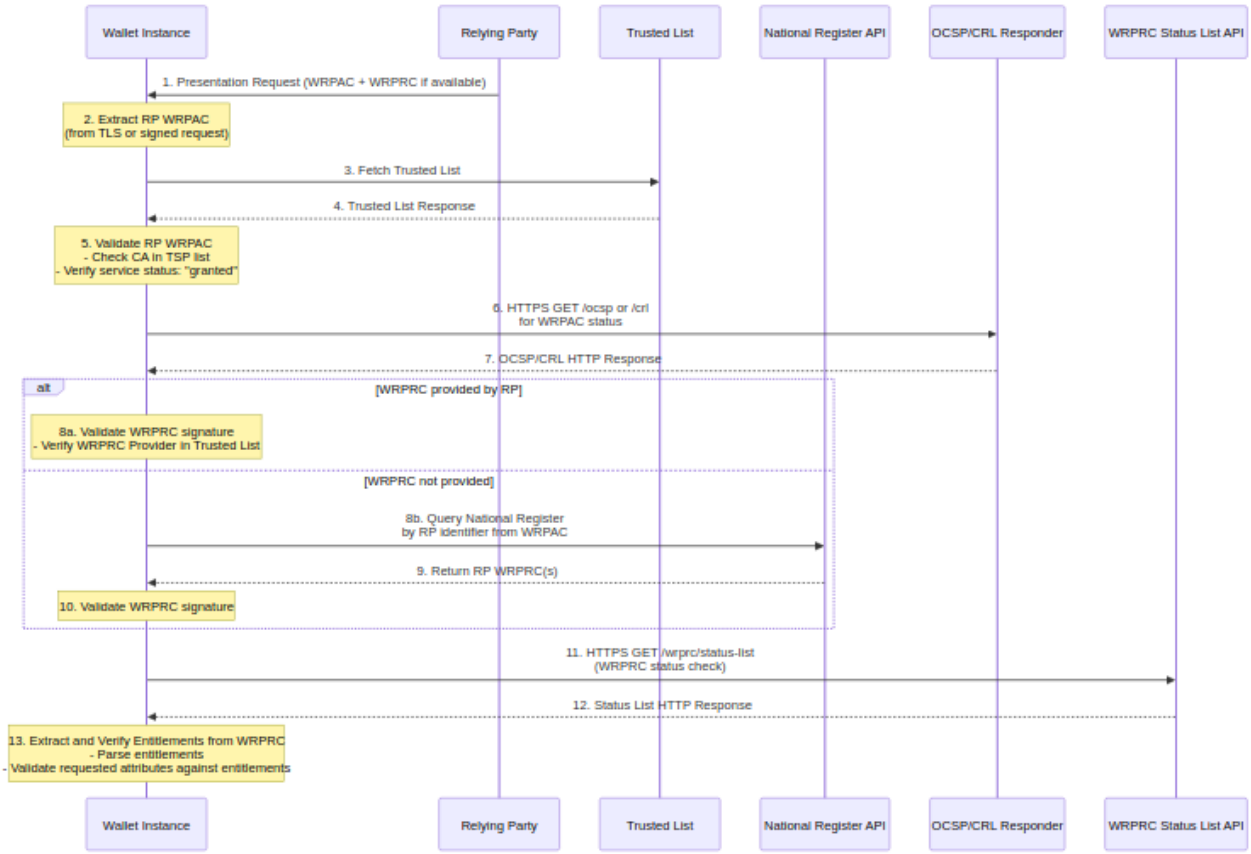
Key lifecycle and Trusted Lists

Key lifecycle for trust anchors and for services listed in Trusted Lists is reflected in list content and status (e.g. service status, status determination approach, and history where required by the profile). Updates and revocation of listed services follow the applicable ETSI trusted list profiles (TS 119 612 / TS 119 602) and are consumed per ETSI TS 119 615. Certificate policies and Certificate Transparency (SCT) where applicable are specified in the referenced ETSI standards.

Relying Party Registration & Access Certificates

Relying Parties (verifiers) **register with the Member State Registrar** before being able to securely identify themselves to Wallet Units. Registration includes identification data, the **attributes** the RP intends to request from Wallet Units, the **intended use** (purpose), and, if applicable, use of intermediaries. The Registrar approves the RP (per ARF Reg_25) and publishes the entry in the **Registry**. The **Access Certificate Authority** then issues access certificates to the RP as described under Certificates and cryptographic anchors, with a **separate access certificate per Relying Party Instance** (Reg_10a). Optionally, the Registrar may request a **registration certificate** from the Provider of Registration Certificates (RPRC_09) that summarises registration status and entitlements. Wallet Units authenticate RPs by validating RP access certificates against the Access CA trust anchors and by verifying registration and requested attributes in the Registry (RPA_04, RPRC_16, RPRC_21). The common API for RP registration information (e.g. TS5) and the common set of RP information (e.g. TS6) are specified in the EUDI Wallet technical specifications. Detailed flows and diagrams are in the WP4 Trust Group trust-infrastructure schema. Certificate issuance aspects are coordinated with the QTSP group.

The following sequence illustrates how a Wallet Instance discovers and validates Relying Party policy during presentation (WRPAC = Relying Party Access Certificate; WRPRC = Relying Party Registration Certificate). Source: WP4 Trust Group, wallet-policy-discovery.



Validation Functions for Relying Parties

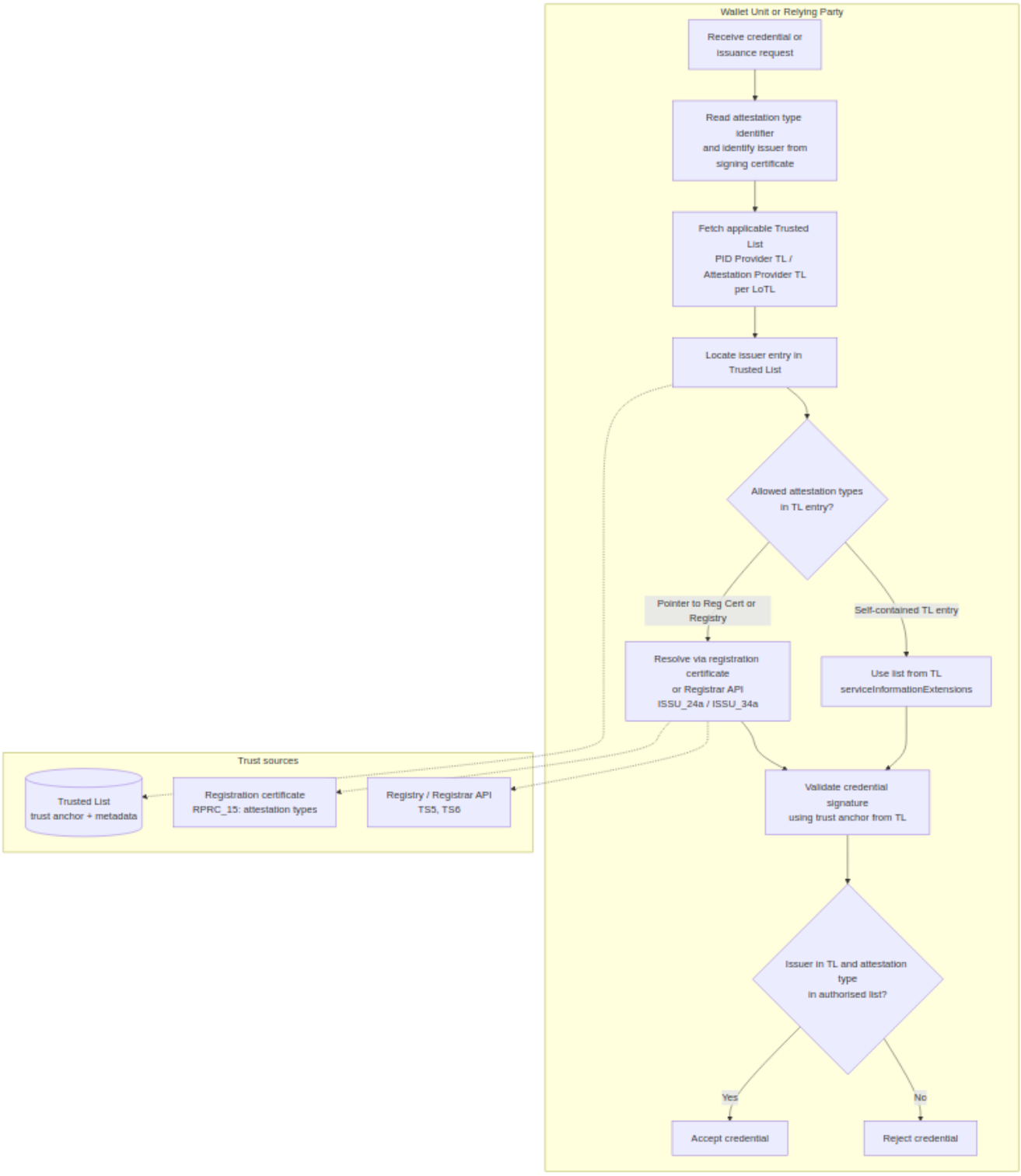
Relying Parties need to **authenticate and validate** the Person Identification Data (PID) and attestations (e.g. LPID/EBWOID, EAA) received from Wallet Units. Validation relies on the trust infrastructure as follows:

- **PID and LPID/EBWOID:** Relying Parties validate the PID (or equivalent) signature using the **List of Trusted Entities (LoTE)** for PID Providers, i.e. the PID Provider Trusted List compiled by the European Commission and referenced from the LoTL. Procedures for authenticating the LoTL and national/EU trusted lists and for obtaining listed services are given in **ETSI TS 119 615**.
- **Attestations (EAA):** Relying Parties validate **qualified EAA (QEAA)** signatures using the **Member State QTSP Trusted Lists** (per Article 22 eIDAS) and **PuB-EAA** (and, where applicable, national non-qualified EAA) using the relevant Attestation Provider Trusted Lists. Trust anchors and service types are defined in ETSI TS 119 602 profiles (e.g. Annex H for Pub-EAA and national EAA provider lists).
- **Wallet and RP side:** Wallet Units verify RPs via the Registry and Access CA Trusted Lists (see Relying Party Registration & Access Certificates). PID Providers and Attestation Providers verify Wallet Providers against the Wallet Provider Trusted List before issuing credentials.

The PID Providers group defines validation semantics for PID/LPID/EBWOID; the QTSP group covers EAA and certificate aspects. Exact validation requirements (e.g. OIA_12, OIA_13, OIA_14) and consumption procedures are documented in the WP4 Trust Group trust-infrastructure schema and ETSI trusted lists implementation profile.

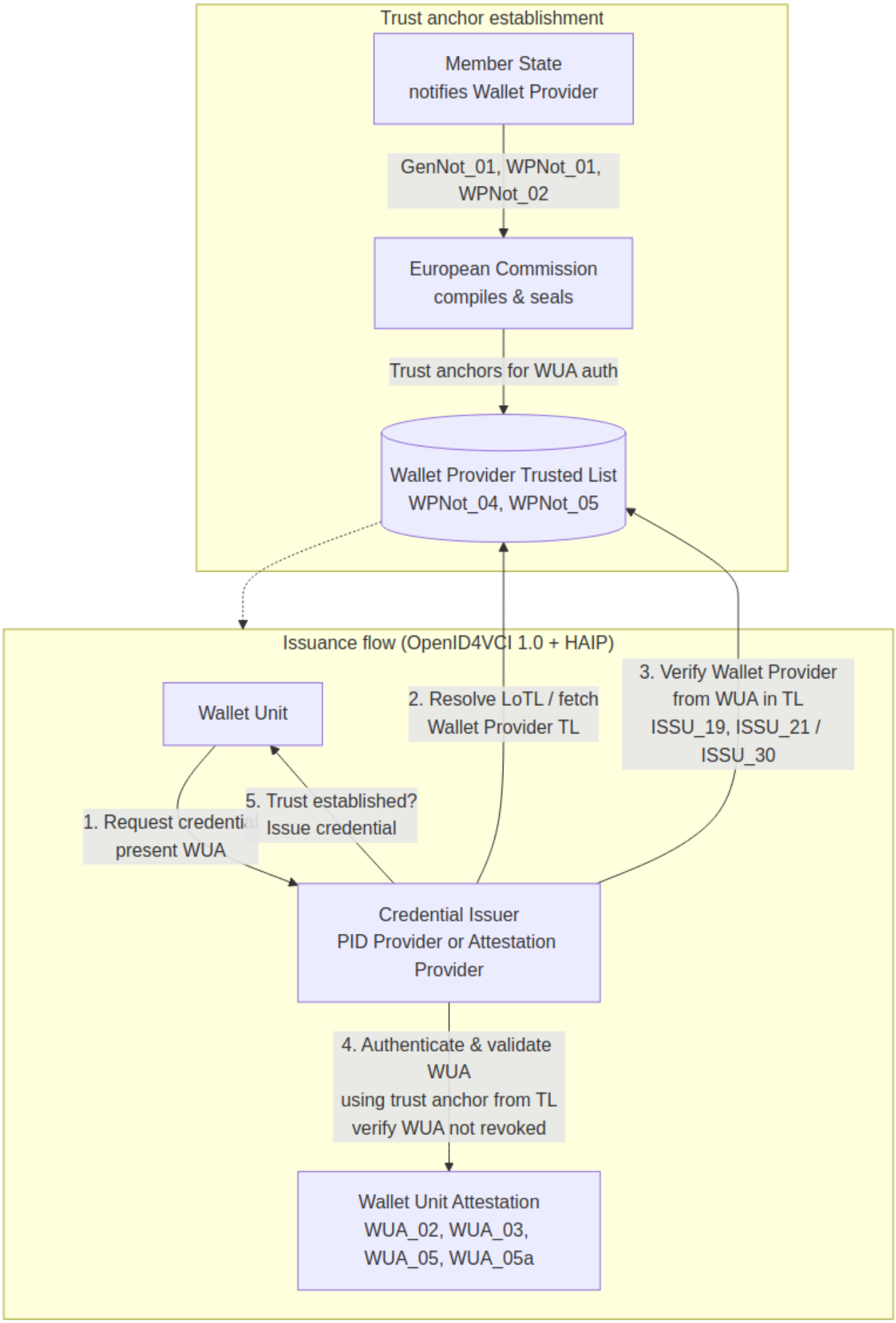
Establishing trust with a Credential Issuer

Wallet Units and Relying Parties establish trust in a Credential Issuer (PID Provider or Attestation Provider) and in the credentials they issue by combining **Trusted List** entries (trust anchors and, where applicable, authorised attestation types) with **registration** data (Registry or registration certificate). The catalogue of attestation schemes defines *what* credential types exist; Trusted Lists and registration define *who* is allowed to issue which types. The flow below is aligned with the WP4 Trust Group credential catalogue and issuer constraints: the verifier accepts a credential only if the issuer is present in the applicable Trusted List and the credential's attestation type is among the issuer's authorised types (ISSU_24, ISSU_24a, ISSU_34, ISSU_34a for Wallet Units; OIA_12, OIA_13, OIA_14, OIA_15 for Relying Parties).



Establishing trust with a Wallet Solution

Trust in a **Wallet Solution** (Wallet Provider and Wallet Unit) is established by **Credential Issuers** (PID Providers and Attestation Providers) before issuing a PID or attestation. The flow is mandated by the ARF (Topic 9 Wallet Unit Attestation, Topic 31 notification and Trusted Lists), uses **OpenID4VCI 1.0** for the issuance protocol and issuer metadata (ISSU_22, ISSU_22a, ISSU_22b), and is reinforced by **OpenID4VC High Assurance Interoperability Profile (HAIP) 1.0** for authenticity, holder authentication, and certificate-bound tokens. The Wallet Unit presents a **Wallet Unit Attestation (WUA)** to the Issuer during the issuance request; the Issuer verifies the Wallet Provider against the **Wallet Provider Trusted List** and validates the WUA (ARF **ISSU_19**, **ISSU_21** for PID; **ISSU_28**, **ISSU_30** for Attestation Providers; **WUA_02**, **WUA_03**, **WUA_05**, **WUA_05a**). The Wallet Provider Trusted List is compiled by the European Commission from Member State notifications (**WPNot_01**--**WPNot_05**).


Trust anchor establishment
Member State
notifies Wallet Provider
GenNot_01, WPNot_01,
WPNot_02
European Commission
compiles & seals
Trust anchors for WUA auth
Wallet Provider Trusted List
WPNot_04, WPNot_05
Issuance flow (OpenID4VCI 1.0 + HAIP)
Wallet Unit
2. Resolve LoTL / fetch
Wallet Provider TL
3. Verify Wallet Provider
from WUA in TL
ISSU_19, ISSU_21 /
ISSU_30
5. Trust established?
Issue credential
1. Request credenti
present WUA
Credential Issuer
PID Provider or Attestation
Provider
4. Authenticate & validate
WUA
using trust anchor from TL
verify WUA not revoked
Wallet Unit Attestation
WUA_02, WUA_03,
WUA_05, WUA_05a

Governance Responsibilities

Trust and security responsibilities are split between **Member States**, the **European Commission**, and **participating entities** as follows:

Responsibility	Owner (normative)	[MVP]	[MVP+]
Registration of PID Providers, Attestation Providers, Relying Parties	Member State Registrar	WE BUILD (mock Registrar)	MS Registrar or WE BUILD
Registry publication and common API (e.g. TS5, TS6)	Member State	WE BUILD (the WP4 reference registry or an other WP4 registry listed in LoTL)	Member State or WE BUILD
Access certificate issuance	Access Certificate Authority (notified by MS to Commission)	WE BUILD (Access CAs listed in LoTL)	Access CA (notified) or WE BUILD (Access CAs listed in LoTL)
Optional registration certificates	Provider of Registration Certificates (notified by MS to Commission)	WE BUILD (WP4 RegCert Providers listed in LoTL)	RegCert Provider (notified) or WE BUILD (WP4 RegCert Providers listed in LoTL)
Compilation and publication of Wallet Provider, PID Provider, Access CA, RegCert Provider Trusted Lists	European Commission	WE BUILD (reference TLs)	European Commission or WE BUILD
Compilation and publication of national EAA Provider TLs and MS QTSP Trusted Lists for QEAA	Member State Trusted List Provider	WE BUILD (mock MS TLP)	MS TLP or WE BUILD
List of Trusted Lists (LoTL), OJEU publication	European Commission	WE BUILD (reference LoTL)	European Commission or WE BUILD
Trust evaluation in Wallet Units and RPs (use of TL/LoTE and Registry per ARF)	Wallet / PID / Attestation / RP implementations	Wallet / PID / Attestation / RP implementations	Wallet / PID / Attestation / RP implementations

Within WE BUILD, the consortium clarifies **which roles are assumed by MVP infrastructure** (e.g. mock or real Registrars, TLPs, Access CAs) and which are assumed to be provided by Member State or EU infrastructure. **Policy and conflicts** (e.g. credential-type or authorization collisions) are handled according to the dispute resolution and collision-prevention mechanisms described in the WP4 Trust Group authentication-authorization and policy framework.

Appendix D. Business Wallet Definition

Revision 1.0

Scope and context

This document sets out a non-technical working definition of “business wallet” as introduced in the European Business Wallet regulatory proposal, to support a common interpretation within WE BUILD and in dialogue with the European Commission. It is intended as reference material for the WE BUILD use case and capability work. It does not cover detailed architecture, protocol choices, implementation design, or use case roadmaps.

This document draws on the EUDI Wallet regulations, EWC deliverables^[1]^, and relevant industry and consortium publications, and incorporates the draft Implementing Act on Business Wallet.

Core concepts

Description

A **Business Wallet** is a product and service that enables an organisation to identify itself, manage authorisations, exchange verified attributes and documents, and receive legally relevant notifications in support of administrative and regulatory procedures. Unlike European Digital Identity Wallets, an European Business Wallet does not need to be an eID means under an eID scheme, although it may reuse similar components.

WE BUILD implementation note: *The topic of online business identity, potentially outside of eID schemes, needs to be further discussed within the WP4 Architecture group. It may also have consequences for the WP4 PID/LPID Providers group.*

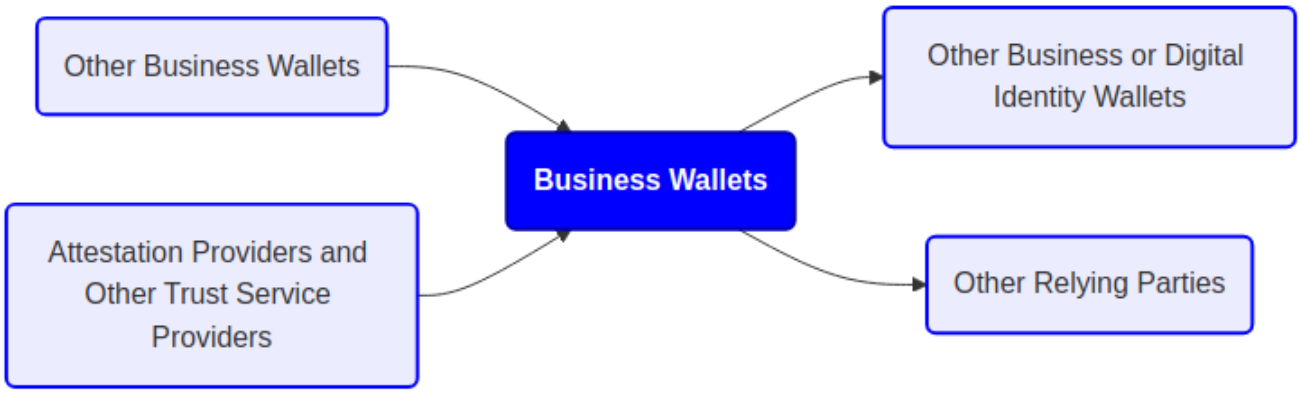
A technical decomposition (front end, back end, and cryptographic components) is out of scope for this document.

Each Business Wallet has a single **wallet owner**, which is the entity that the wallet represents through its interactions. Note that this is distinct from, for example, the company owner or the wallet provider.

The wallet owner is defined by **European Business Wallet Owner Identification Data (EBW-OID)**, which includes an official name and an EU-unique identifier. These owner identification data are issued into the business wallet as an electronic attestation of attributes.

A Business Wallet can have multiple **wallet users**, meaning natural or legal persons that operate the wallet through a user interface or an application programming interface under roles and mandates set by the wallet owner. These wallet users may apply software applications to access these interfaces. Some users may be **authorised representatives**, while others may be employees or service providers operating within delegated permissions.

Conceptual model



Business Wallet definition

Roles supported

A business wallet enables its owner, amongst other operations, to act as:

- Issuer, holder or verifier of electronic attestations of attributes
- Signatory or origin of sealed data
- Sender or recipient of messages, such as submissions and notifications

These operations are under role-based access control, where recognised roles comprise:

- Wallet owner: the entity that is accountable for the legal consequences of the operation
- Authorised representative: a wallet user with an administrative mandate to act on behalf of the wallet owner, potentially with a limited scope or in limited contexts

In addition, the wallet owner may configure other roles that suit the owner's policies and/or national or EU law.

Other relevant roles are:

- Wallet provider: the entity that provides the business wallet solution to its owner (potentially the owner themselves)
- Owner identification data provider: the entity that verifies the identity of an authorised representative enrolling the wallet owner and attests, using an electronic attestation of attributes, the wallet owner's identification data in accordance with authentic source registrations

Key functions

Wallet lifecycle management

The business wallet enrols its owner via the electronic identification of an authorised representative and facilitates enrolment in connected trust services and directory services. The wallet provider is responsible for attesting to its validity to relying parties and enabling authorised

representatives to revoke the business wallet and perform other lifecycle changes. In several cases, the wallet provider is also responsible for notifying authorised representatives and government authorities about lifecycle changes.

WE BUILD implementation note: *This will be the responsibility of the WP4 Wallet Providers group. At least several providers will be ready to manage their wallet solution and issue wallet units under new and changing business wallet requirements.*

Digital document management

The business wallet enables the wallet owner to create, store, use and validate various types of digital documents:

- Electronic attestations of attributes (EAAs, including QEAA, PuB-EAA, and EAA issued by the Commission)
- Business documents, such as electronic invoices
- Qualified certificates for electronic signatures and seals
- Qualified electronic signatures, seals and timestamps
- Evidence, such as provided by trust service providers upon electronic transactions, or by public sector bodies over the single digital gateway

For this purpose, the business wallet implements several applications, including signature creation and secure cryptographic applications.

WE BUILD implementation note: *The WP4 Wallet Providers provide, as part of their business wallet solutions, a subset of the functionalities required by the use cases. For the functionalities that require qualified trust services, such as the issuance of qualified certificates or the sealing of documents with qualified electronic seals, the WP4 QTSP group provides these services within WE BUILD. For reference, see the QTSP documentation.*

Secure communication channel

To enable public and private sector information exchange, such as in B2G eGovernment notifications, B2B/B2G eProcurement business documents and other business use cases, a business wallet implements a secure communication channel with other business wallets, with users of digital identity wallets, or with alternative solutions provided through a gateway. This channel enables cross-border delivery and receipt of submissions and notifications with legal effect, and provides a trusted channel with public authorities and other regulated parties across the EU. The channel is implemented using a qualified electronic registered delivery service (QERDS). The digital address for the channel is registered in a standard digital directory.

WE BUILD implementation note: *the WP4 QTSP group will explore delivering an interoperable pre-production QERDS, along with CIR (EU) 2025/1944 requirements, as a service to the WP4 Wallet Providers group, working with the WP4 Architecture group on cross-cutting concerns, such as interoperability specifications. This enables wallet providers to provide a business wallet to the use cases with a digital address and access to the designated QERDS. For reference, see the QERDS documentation.*

Access control mechanism

To enable wallet owners, authorised representatives and other authorised users to access the business wallet while preventing unauthorised access, each business wallet implements role-based access control for the assets it protects, including digital documents and the secure communication channel. To identify, authenticate, and authorise wallet users, the access control mechanism relies on electronic identification means, such as digital identity wallets, and, potentially, on trust services for the electronic attestation of attributes.

WE BUILD implementation note: *The WP4 Architecture group, in collaboration with the WP4 Wallet Providers group, will explore the access-control mechanism for business-wallet solutions. This may rely on the EUDI wallets within WE BUILD or on other electronic identification means.*

Digital transaction management

Business wallets keep logs and provide dashboard user interfaces to enable control over transactions, including operations on the wallet lifecycle and on digital documents and messages sent and received over the secure communication channel. In addition, these logs enable dispute resolution regarding potentially unauthorised transactions, failures to meet reporting obligations, or administrative or procedural activities.

Appendix E. Wallet Implementation and Deployment Considerations in WE BUILD

This appendix provides a short overview of wallet implementation and deployment approaches observed among WE BUILD wallet providers. It does not repeat the architectural classifications defined in the ARF, but highlights aspects that are relevant for the WE BUILD pilots.

Different wallet implementations exist in the ecosystem, reflecting different user groups, device capabilities, and deployment environments. In practice, implementations often combine different approaches depending on the supported use cases.

Wallet Types Relevant for WE BUILD

From a deployment perspective, wallet implementations in the WE BUILD ecosystem can broadly be grouped into four practical categories.

Wallet Type	Typical Deployment	Primary Use Cases
Mobile (on-device)	Smartphone application using device hardware security	Natural person wallets and offline use cases
Web / browser-based	Browser interface with backend cryptographic services	Desktop services and enterprise workflows
Cloud / HSM-based	Server-hosted wallet infrastructure backed by HSMs	Legal person wallets and managed services
Hybrid	Combination of local device security and remote HSM	Mixed use cases requiring both scalability and offline capability

These categories reflect common deployment patterns observed across wallet implementations. The concrete architecture used by a wallet provider depends on the supported use cases, operational requirements, and device capabilities.

Deployment Patterns Observed Among WE BUILD Wallet Providers

The WE BUILD Wallet Provider Group conducted a stocktaking questionnaire covering **31 wallet providers** participating in the project. Providers described the deployment models they currently support.

The results show a clear split between natural person and enterprise wallet deployments.

Deployment Option	Share of Providers
Mobile wallet (iOS/Android app)	77%
Server wallet on cloud	55%
Server wallet on-premise	42%
Multi-device or white-label wallet	6%
Wallet functionality via API or SDK	6%

Many providers support multiple modes, typically combining a mobile wallet for natural persons, and a cloud or server-based wallet for legal persons.

Architectural Trends in the WE BUILD Ecosystem

The stocktaking exercise highlights several trends relevant for the WE BUILD pilots.

Mobile and cloud duality

The most common architecture combines:

- a **mobile wallet for natural persons**, and
- a **server-based wallet for enterprise or legal person scenarios**.

This reflects the broader EUDI ecosystem, where personal identity use cases are mobile-centric while organisational use cases often require backend infrastructure.

Increasing use of HSM-backed infrastructure

Several providers indicate the use of remote HSM infrastructure for enterprise wallet deployments. This approach supports large-scale operations and key recovery but requires continuous network connectivity.

Limited visibility of WSCD implementation choices

The questionnaire responses mainly describe the application layer (mobile app, server, or web wallet), rather than the underlying cryptographic architecture.

Only a small number of providers explicitly describe the type of secure cryptographic device used

(for example secure hardware on the device or remote HSM infrastructure).

Emerging architectures for legal person wallets

Architectures supporting legal person wallets are still evolving.
Many providers indicate that their legal person wallet solutions will be further developed during the WE BUILD project in alignment with emerging European Business Wallet proposals.

As a result, the architectures described in the stocktaking responses should be understood as initial implementation approaches rather than final designs.

Appendix F. QTSP documentation

The WP4 QTSP group collaborates on internal reference code and documentation to increase interoperability. This appendix lists the entry points for this reference documentation by each provided service.

QES documentation

This is documentation of the WE BUILD: WP4 QTSP group.

Technical Overview of Signing and Sealing in WE BUILD: Signing and Sealing

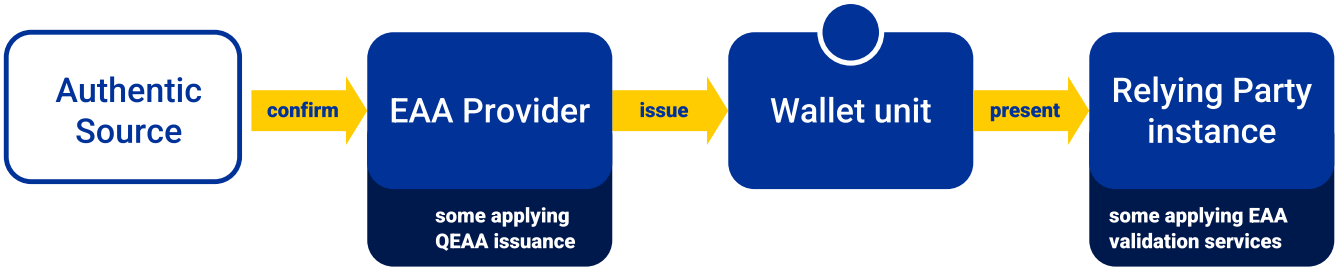
Informative references

- Standards
  - Stable
    - CSC API v2.2.0.0
    - CSC DM v1.0.0
    - CSC_data-model-bindings v1.0.0
  - Work items
    - eudi-doc-standards-and-technical-specifications#29: CSC standards updates
    - eudi-doc-standards-and-technical-specifications#68: TS 119 432 update
- Reference specifications
  - German EUDI blueprint: QES

QEAA documentation

This is part of the QTSP documentation.

Reference model



Architecture overview

- Architecture overview for QEAA in WE BUILD

Feature definitions

Below is a non-exhaustive overview of QEAA features that use cases may choose to pilot. For each feature in scope for the pilots, the QTSP group develops an interop profile and ensures available service compatibility.

- QEAA issuance to EUDIW
- EAA validation at RP
- Verification of attributes

Schemes for QEAA

Participants of the QTSP group may issue QEAA under any of the schemes referenced below.

- Hello World Attestation

Informative references

- Catalogues
  - Attestation Rulebooks Catalog: Catalogue of schemes for EAA in EUDI
- Standards
  - TS 119 471 v1.1.1: requirements for EAA Providers
  - TS 119 472-1: **DRAFT** Profiles for EAA - General requirements
  - TS 119 472-2: **DRAFT** Profiles for Relying Party Interface to EUDI Wallet
  - TS 119 472-3: **DRAFT** Protocol Profiles for interfacing to services providing Personal Identity Data and Electronic Attestation of Attributes
  - TS 119 612: Policy and security requirements for trust service providers issuing electronic attestation of attributes (EAA)
  - TS 119 602: Electronic signatures and infrastructures (ESI); Policy and security requirements for trust service providers issuing attribute attestations
  - ETSI TS 119 478: **DRAFT** Protocol Interface for Trust Service Provider use of Authentic Sources
- Technical reports

- TR 119 476 v1.2.1: SD and ZKP for EAA analysis
- TR 119 476-1 v1.3.1: SD and ZKP for EAA feasibility
- TR 119 479-2 v1.1.1: EAA extended validation

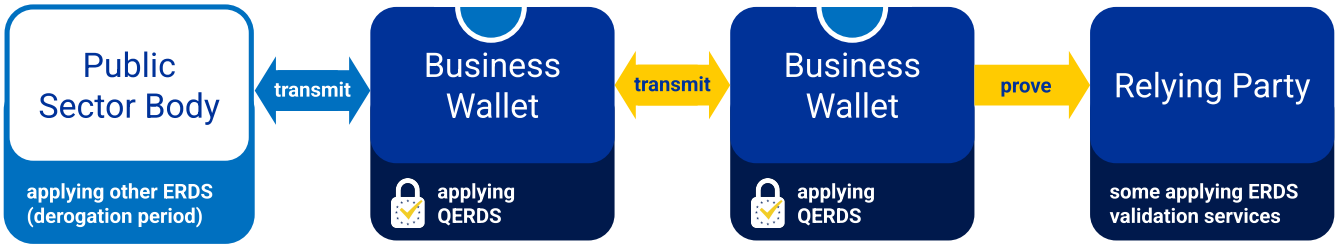
QERDS documentation

This is documentation of the WE BUILD: WP4 QTSP group.

Scope:

- Provision of electronic registered delivery services

Reference model



Architecture overview

- Architecture overview for QERDS in WE BUILD

Feature definitions

Below is a non-exhaustive overview of QERDS features that use cases may choose to pilot. For each feature in scope for the pilots, the QTSP group develops an interop profile and ensures available service compatibility.

- QERDS between wallets

Technical reports

- QERDS interoperability framework requirements

Informative references

None yet.

rWSCD documentation

This is documentation of the WE BUILD: WP4 QTSP group. Scope:

- Management of remote wallet secure cryptographic devices

Informative references

None yet.

RPAC/RPRC documentation

This is documentation of the WE BUILD: WP4 QTSP group. Scope:

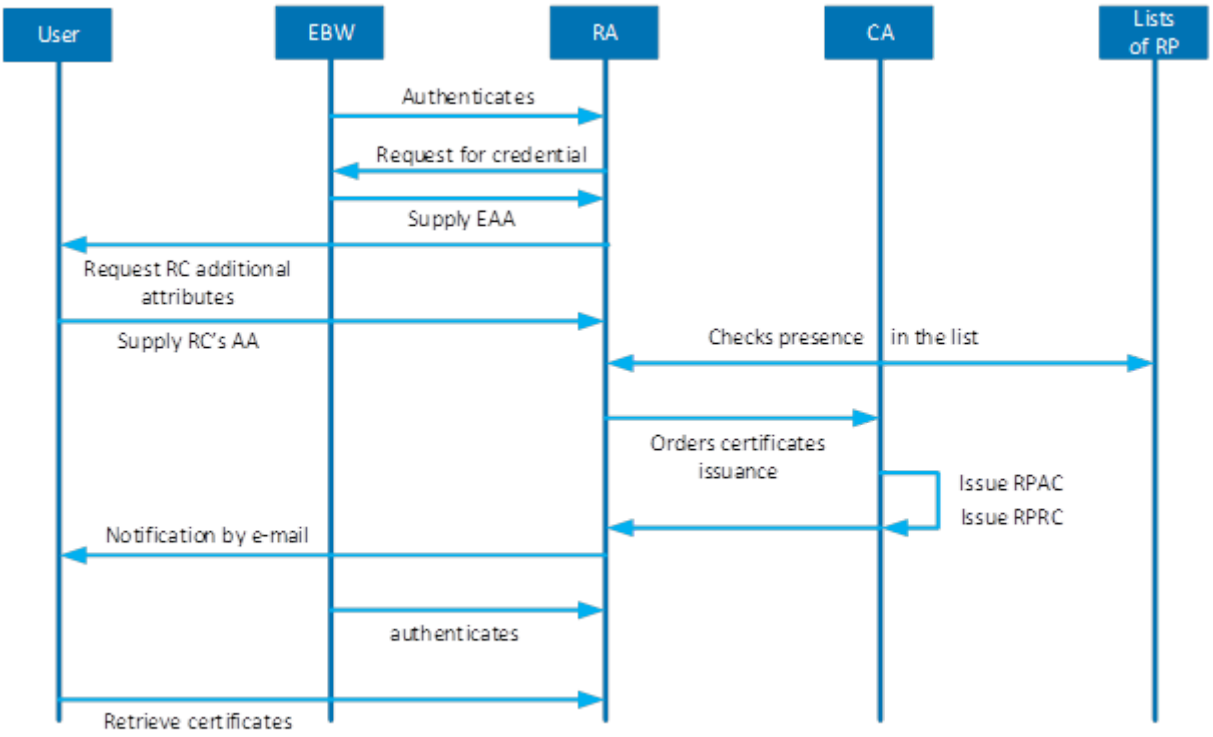
- Relying party access certificate issuance
- Relying party registration certificate issuance This is documentation of the WE BUILD: WP4 QTSP group. Scope:
  - Relying party access certificate issuance
  - Relying party registration certificate issuance

Assumptions:

- In complete eIDAS ecosystem, Registrars will act as Registration Authorities for TSP issuing RPAC & RPRC. Without any actual Registrar included in the project, present RAs of involved TSPs shall play the role of Registrar.
- Registrars did not publish Registration policies yet. Considering WE BUILD as a close group where trust comes from belonging, registration policy shall rely on checking this belonging together with a public NCP registration policy for RPAC: TSP will check presence in lists of WP's RP in place of national register of Wallet Relying Parties. WP's leader shall establish these lists.

Issuance process: Process of issuance will limit to AnnexD1 of TS 119 475 for MVP. It will be made of 11 steps :

1. The user (RP representative) will connect and authenticate to TSP's RA by using its EBW;
2. RA will request credentials;
3. And receive as an answer a EAA granting the user to act as representative of the RP (POA);
4. RA contacts the user to request all additional attributes needed for producing RPRC;
5. RA checks that the authenticated RP is present in the lists of authorised RP supplied by WP's leaders;
6. RA orders issuance of both certificates;
7. CA issues RPAC and RPRC;
8. CA transmits certificates to the RA;
9. RA notifies the user (e.g. by e-mail);
10. User authenticates via EBW;
11. User retrieves RPAC & RPRC.



Informative references

Standards:

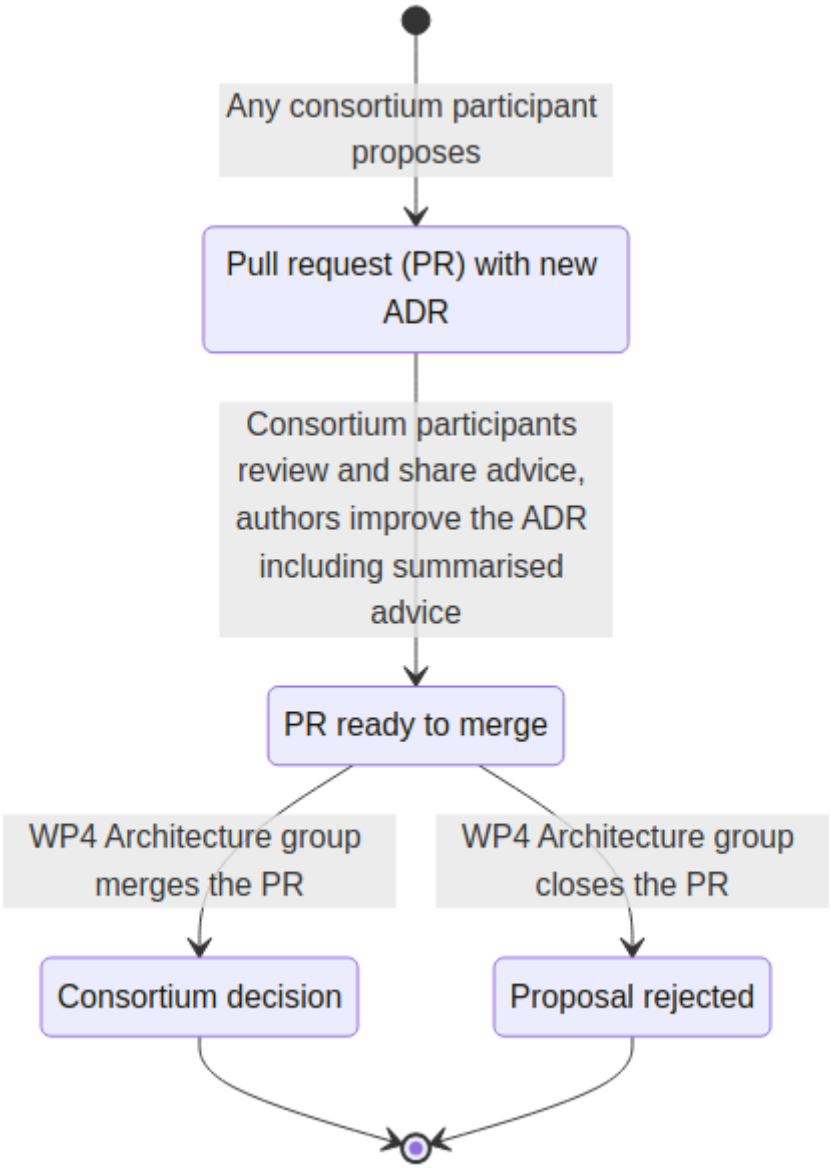
- EN 319 401
- EN 319 411-1&2
- TS 119 475
- TS 119 411-8
- IETF 7515 Technical Reports:
- TR 119 411-9

Appendix G. Architecture decision records

WE BUILD maintains a lightweight architecture decision record (ADR) for each software-related decision affecting interoperability.

Propose new ADRs using the template. Announce them to the Architecture group in the Portal to get feedback to understand the consortium's opinion.

ADR process for WE BUILD



ADR overview

Publish consortium trusted lists

Authors:

- Sander Dijkhuis, Cleverbase, the Netherlands

Context

For the first usage of the interoperability testbed, and for the first increment, it is required to enable trust between issuers, wallets, and verifiers. EWC proposed a trust evaluation mechanism (see EWC RFC 012) including an Trusted List.

According to the evaluation made so far by the Mapping Task Force, at least the QTSP group needs a similar approach for the first QEAAs to test. We need to specify who provides this.

At ETSI, the TS 119 612 (V2.3.1 at time of this release) series is being updated to support the European Digital Identity. This may require further alignment with the current EWC

implementations.

Decision

The WP4 Trust Registry Infrastructure group provides trusted lists.

The issuers in the WP4 PID/LPID/QTSP/Wallet provider groups request registration on the trusted lists before testing.

The wallets in the WP4 Wallet provider group include validation with the trusted lists in their verification processes.

The relying parties in WP2/3/4 include validation with the trusted lists in their verification processes.

The starting point is TS 119 612 V2.3.1. Since a profile and implementation guidance are likely needed, and EWC RFC 012 has not yet been effective in practice, these should be specified in separate WE BUILD architecture documents.

Consequences

This decision makes interop between consortium members easier.

This decision makes it harder to test with ad-hoc pairwise trust relationships.

To address the risk of bottlenecks in implementing this decision, the Mapping Task Force pays extra attention to potential blockers for each involved WP4 group.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heard the following advice.

- 2025-10-27, Leif Johansson, Sunet, Sweden: OK
- 2025-10-27, Giuseppe De Marco, Dipartimento per la trasformazione digitale, Italy: OK

Baseline protocols

Authors:

- Leif Johansson, Sunet, Sweden
- Sander Dijkhuis, Cleverbase, the Netherlands
- Sarah Amandusson, Digg, Sweden
- George J Padayatti, iGrant.io, Sweden
- Lal Chandran, iGrant.io, Sweden

Context

The EUDI Wallet ecosystem is mandated by eIDAS to ensure secure, interoperable cross-border

digital identity. This implementation is guided by the Architecture and Reference Framework (ARF), which translates legal mandates into technical specifications. To achieve mandatory interoperability for Issuance and Presentation of Person Identification Data (PID) and Electronic Attestations of Attributes (EAA), implementing regulations require compliance with foundational standards. OID4VCI (Issuance) and OID4VP (Presentation) are adopted as the ARF-recommended baseline protocols for technically implementing the required data models and web-based transport mechanisms in the pilot.

Proximity flows are out of scope for the current architecture decision.

While the architecture decision to Publish consortium trusted lists based on TS 119 612 has already been recorded, the consortium also needs a selection of PID/EAA issuance and presentation protocols.

Decision

Each recognized role in the WE BUILD project - PID/LPID Providers, EAA Providers (including QEAA, Pub-EAA), Relying Parties, Wallet Providers, and Trust Service Providers — is REQUIRED to implement the corresponding technical profiles described here. Actors performing multiple roles MUST meet all requirements relevant to those roles.

- PID/LPID Providers, EAA Providers (including QEAA, Pub-EAA) MUST implement OpenID4VCI version 1.0
- Relying Parties MUST implement OpenID4VP version 1.0
- Wallet Providers MUST implement in wallet solutions OpenID4VCI version 1.0 and OpenID4VP version 1.0

Consequences

Implementations following this profile ensure interoperability between actors within the WE BUILD ecosystem. Actors can verify conformance through testing against other implementations.

The selected protocols may not suffice for async or proximity use cases. Once the requirements for such use cases appear, the decision may need to be nuanced to enable for example Relying Parties to implement different protocols.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision.

Specify PID and eAA formats

Authors:

- Sander Dijkhuis, Cleverbase, the Netherlands

Context

To issue PID (including LPID) and eAA (including QeAA and PuB-EAA), it is necessary to specify a digital document format such as mdoc or a JWT-based one. While the EUDI implementing acts specify baseline standards, these specifications may not suffice for WE BUILD. For example, some specified standards need further profiling for interoperability. For some WE BUILD use cases, other standards may be necessary, for example to enable asynchronous interactions with EU Business Wallets, or to enable legal, technical and semantic interoperability with existing systems.

Several WP4 groups are stakeholders in this, for example:

- WP4 Architecture, for conformance to and efficient implementation of the EUDI framework
- WP4 PID and LPID Providers, for technical details of the provision of PID and LPID
- WP4 QTSPs, for technical details of the issuance and validation of QeAA
- WP4 Semantics, where the specification affects semantic interoperability

To ensure a streamlined process, the Mapping Task Force has evaluated the possible dependencies within WP4.

Out of scope of this evaluation were the dependencies with WP2 and WP3. After the use case stock taking, these should guide the decisions about formats made within WP4. If needed, the WP4 Architecture support teams can help gain input and feedback.

Decision

The WP4 Architecture group specifies digital document formats for PID and eAA.

The WP4 PID and LPID Providers and the WP4 QTSPs are expected to conform to these decisions.

The vocabularies that WP4 Semantics group delivers, may be in a format that is unrelated to the specified digital document formats.

The WP4 Semantics group is expected to ensure semantic interoperability of digital documents using the specified digital document type, if the specified digital document type supports semantic mapping, such as in Verifiable Credentials Data Model v2.0. If not, such as in ISO/IEC 18013-5 mdoc, extra translation may be needed to match vocabularies specified by the WP4 Semantics group. In such cases, the translation is the responsibility of the scheme owners.

Consequences

This decision clarifies the decision process regarding digital document formats for PID and eAA. Other decisions should be recorded soon after to specify the concrete formats for PID and eAA for European Digital Identity Wallets and for European Business Wallets.

This decision creates an indirection between the vocabularies and the digital document formats for PID and eAA, so that the extra translation may be needed if the format does not support semantic annotations. In this case, the translation is the responsibility of the scheme owners.

As an alternative, it has been considered to decide on WP4 Semantics specifying digital document formats for PID and eAA, which would facilitate semantic interoperability, potentially at the cost of

technical and legal interoperability. However, since these are cross-cutting decisions, the Mapping Task Force suggests that instead the WP4 Semantics group advices the WP4 Architecture group so that the latter sufficiently takes semantic interoperability into account, for example the ability to support links to globally defined semantic models.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heared the following advice.

- 2025-11-11, Ronald Koenig, Spherity, Germany: OK

Provide EBWOID as a stable minimal basis

Authors:

- Sander Dijkhuis, Cleverbase, the Netherlands

Context

In BU3 on foreign tax declaration, an issue on Identifiers in LPID was raised to WP4 Architecture, which extends to EBWOID. In this context, EBWOID is *European Business Wallet owner identification data*. Should the EBWOID just be a "bootstrap identity" with a stable minimum attribute set, or should EBWOID be a "dynamic reference framework" containing many relevant attributes registered by competent bodies?

In the Annex of (EU) 2024/2977, Table 3 specifies mandatory legal person identification data in line with the "bootstrap identity", and Table 4 specifies optional legal person identification data that leans more towards the "dynamic reference framework".

In COM(2025) 838 final, Article 8(5) the EBWOID is specified to contain just the name and unique identifier in accordance with Article 9. Furthermore, in Article 20, legal person identification data may become irrelevant under EU Digital Identity.

The implementation choice affects what other electronic attestations of attributes may be needed for use cases such as BU3.

Under EU Digital Identity, EU Member States may take different decisions with regard to this. The upcoming EU Business Wallet legislation may affect these decisions.

To achieve cross-border interoperability within WE BUILD, several options are possible:

- basic EBWOID everywhere (minimum attributes from a single source)
- extended EBWOID everywhere (attributes such as the VAT registration number)
- basic EBWOID in some EU Member States, extended EBWOID in others

Decision

Rely on basic EBWOID everywhere as a minimum identity attestation which must be supported by everyone. Develop use cases under the assumption that other attributes require additional

electronic attestations. These other attributes can be used both for identifying the economic operator and for verifying additional claims.

Consequences

The EBWOID rulebook should be kept in line with this decision.

With this decision, it becomes easier to reason about the minimum set of additional electronic attestations.

With this decision, it becomes more difficult to test the extended EBWOID case, which may be relevant to some EU Member States. But note that the decision does not preclude testing with extended EBWOID as well.

To manage the risk that this approach differs from EU Business Wallet legislation, WE BUILD should take the definition of EBWOID in account in its upcoming definition of an EU Business Wallet.

To get started with the minimum identification data, the WP4 PID/LPID group should specify which unique identifier(s) to use.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heard the following advice.

- 2025-11-17, Michelle Ludovici, Digg, Sweden: OK, but this does not yet solve the question: which unique identifier should be used in the LPID and EBWOID? Proposing to use the European Unique Identifier from (EU) 2017/1132.
- 2025-11-18, Ronald Koenig, Spherity, Germany: Only acceptable if it does not preclude the use of more comprehensive "identity attestations" (EUCC, KYC, etc.).
- 2026-02-02, Erwin Nieuwlaar, KvK, the Netherlands: The PID/EBWOID group agrees the EBWOID should consist of a minimal and stable set of fields. There is still discussion about whether EBWOID is necessary for a functional EBW (out of scope for this ADR).
- 2026-02-03, Jonas Toennis, Brønnøysund Registry Center, Norway: OK, and note that the PID/EBWOID group considers the EBWOID to functionally have the same effect for business wallets as PID for digital identity wallets. Also, there are questions regarding the use of other attestations for company identity (out of scope for this ADR).
- 2026-02-09, Sarah Amandusson, Digg, Sweden: Note the alignment on LPID removal and EBWOID spelling from pending ADR #67.

Wallet Unit Attestation and Lifecycle Management (For European Business Wallet)

Status: Proposed
Date: 11 February 2026
Authors: WP4 Architecture

- Lal Chandran, iGrant.io, Sweden
- Sander Dijkhuis, Cleverbase, the Netherlands
- George J Padayatti, iGrant.io, Sweden

Context

The European Business Wallet represents an economic operator or public sector body and operates within a cloud or on-premise environment under regulatory oversight derived from revised eIDAS and related Implementing Regulations.

Trust in a Business Wallet must be established at two distinct levels:

1. Cybersecurity assurance in the wallet's secure execution and key management environment.
2. Organisational entity authentication assurance (cf. ISO/IEC 29115) represented by the European Business Wallet Owner ID (EBWOID).

At present, the WE BUILD architecture does not formally define:

- A Wallet Unit Attestation (WUA) model for European Business Wallet.
- A Wallet Instance Attestation (WIA) model and its relationship to the WUA.
- A clear lifecycle model governing Wallet Unit states.
- The downgrade and revocation semantics between structural and identity trust.

Without an explicit lifecycle and attestation model, revocation handling, issuance eligibility and cross-border interoperability remain ambiguous.

Decision

The European Business Wallet SHALL introduce:

1. A mandatory Wallet Unit Attestation (WUA).
2. A defined lifecycle model governing Wallet Unit state transitions.
3. For this ADR, WIA is not considered for the first iteration of WE BUILD at least.

Wallet Unit Attestation (WUA)

A Wallet Unit Attestation is a signed object issued by the Wallet Provider that:

- Binds the Wallet Unit to a secure cryptographic environment.
- Contains public keys used for credential binding.
- Includes validity and revocation information.
- Is presented to Issuers and Attestation Providers.
- Is presented to Relying Parties when required to determine Wallet Unit validity, for example via selective disclosure mechanisms.

A valid WUA is required for a Wallet Unit to operate within the EBW ecosystem.

Wallet Unit Lifecycle Model

The Wallet Unit SHALL follow the lifecycle states defined below:

UNINSTALLED

No wallet instantiated. No cryptographic material. No WUA. No EBWOID.

INSTALLED

Wallet software deployed and environment prepared, but no WUA issued.

OPERATIONAL

Valid WUA present. Structural trust established. EBWOID not yet acquired or pending.

VALID

Valid WUA and valid EBWOID present. Fully functional for regulated and cross-border use.

State Transitions

- **UNINSTALLED → INSTALLED**
  Organisation instantiates the software.
- **INSTALLED → OPERATIONAL**
  Wallet Unit acquires the WUA.
- **OPERATIONAL → VALID**
  A natural person requests and obtains the EBWOID on behalf of the Economic Operator, which is then bound to the Wallet Unit.
- **VALID → OPERATIONAL**
  EBWOID revoked or expired.
- **OPERATIONAL or VALID → INSTALLED**
  WUA revoked or expired.
- **INSTALLED → UNINSTALLED** Uninstallation or decommissioning of a WU, for e.g. during porting from one service provider to other.

Structural trust (WUA) is foundational. Identity trust (EBWOID) depends upon it. A Wallet Unit cannot be VALID without a valid WUA.

Consequences

- Cybersecurity assurance and Organisational entity authentication assurance are explicitly separated.
- Revocation of WUA immediately suspends infrastructural legitimacy.
- Revocation of EBWOID suspends identity validity while preserving structural trust. Where EBWOID is short-lived, the dependency between WUA revocation and EBWOID validity must be further specified in the implementing acts to ensure Relying Parties are not exposed to invalid Wallet Units.

- Issuers gain a clear rule for issuance eligibility.
- Lifecycle handling becomes testable within ITB and conformance specifications.
- Cross-border interoperability is strengthened through explicit state semantics.
- Wallet Providers are expected to implement this which will be elaborated on further, for.g. via a conformance specification.

This ADR establishes WUA and lifecycle management as mandatory architectural functions of the European Business Wallet. It does not yet establish lifecycle management for entry in the WE BUILD Digital Directory, to simulate the European Digital Directory. This is another layer, which governs the availability of the wallet owner for notifications and submission of documents.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heard the following advice.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heard the following advice.

2026-02-11, Sander Dijkhuis, Cleverbase, Netherlands 2026-02-11, George Padayatti, iGrant.io, Sweden 2026-02-12, George Fourtounis, GRNET, Greece 2026-02-12, Malin Norlander, Bolagsverket, Sweden

Deliver business wallet data using QERDS

Authors:

- Sander Dijkhuis, Cleverbase, the Netherlands
- Leif Johansson, SIROS, Sweden

Context

The WP4 Architecture group discussed at the IRL Workshop the draft on eDelivery for wallet-to-wallet messaging. Afterwards, the European Commission published the European Business Wallet (EBW) proposal COM(2025) 838 final that includes a secure communication channel: a designated qualified electronic registered delivery service (QERDS). See the definition in Business Wallet Definition (version 2026-01-14). This definition and the WE BUILD approach was discussed during the Business Wallet Workshop.

While the QERDS was already part of the WE BUILD Grant Agreement, this record makes it explicit part of the WE BUILD architecture and specifies high-level starting points. Use cases are encouraged to specify scenarios with the QERDS in wallet-to-wallet interactions as well as using gateways to connect to existing infrastructures.

The QERDS should comply to the eIDAS requirements for QERDS as well as the proposed EBW requirements for the designated QERDS. The Commission Implementing Regulation (EU) 2025/1944

applies regarding “reference standards for processes for sending and receiving data in [QERDS]s and as regards interoperability of those services”. For the designated QERDS, no draft implementing act is available yet, but high-level requirements are included in the EBW proposal Annex chapter 11, as well as Article 5(3) for availability to European Digital Identity Wallets. It is expected that WE BUILD implementation experience can contribute to the specification of the EBW implementing act.

Key assumptions are:

- the use of the European Digital Directory (EBW Article 10) for identification (looking up EUIDs), discovery (finding networks and capabilities), and connections (retrieving protocol and endpoint information including public keys);
- the ability for multiple qualified trust service providers to federatively provide the QERDS;
- the application of architecture models from the reference standard EN 319 522-1:
  - 4-corner model (§ 4.3) for EBW-to-EBW exchange;
  - extended model (§ 4.4) for exchange through a gateway (EBW Article 16(6)(b)).

For EBW-to-QERDS communication, there does not yet seem to exist a common protocol and the WP4 Architecture group has discussed various ideas.

For communication between QERDS providers, the reference standard EN 319 522-4-1 specifies bindings based on AS4 (HTTP-based protocol in ISO 15000-2:2021, OASIS Standard) or email. The AS4 standard is referenced in EU legislation and implementations such as Peppol using the eDelivery AS4 building block. Other protocols that follow the same architectural model exist, and are outlined in the considered alternatives later in this section.

The AS4 protocol is based on XML signature and encryption which does do not yet provide post-quantum safety. There is no current effort in any standards development organisation to propose post-quantum algorithms for XML signatures or key agreement/encryption. This means that the effective lifetime of a solution based on AS4 as-is is limited to a maximum of 6 or 7 years from the required go-live date for the European Business Wallet. Therefore, in a production ecosystem, protocol migration should be possible.

The 4-corner model of the AS4 architecture provides a way to introduce an abstraction layer towards the QERDS, which means that in principle it is possible to replace the underlying QERDS protocol in the future, although such a task would present significant challenges in practice.

The following alternatives were considered:

- OpenID4VC and ISO/IEC 18013-7 are standards for receiving credentials into and generating and presenting proofs from the EUDI Wallet. These protocols are well suited for flows that involve user interaction, but are more difficult to adapt to flows that involve automated systems or agents.
- DIDComm is conceptually quite similar to AS4 and has many benefits including a better path towards quantum safe signatures and encryption than an XML-based protocol has at this point. The downside of this alternative is that the standards need to be profiled for use in the EU, for example to reference existing trust infrastructure.
- Alternatively, a new suite of protocols could be developed that fulfill the expected requirements

of the EUBW such as suitability for automation and agents, compatibility with the EUDI natural person wallet etc. There are several options that could serve as a modern starting point for such work including the Matrix protocol and ActivityPub.

Decision

WE BUILD tests and pilots a designated QERDS, with the following responsibilities:

- WP4 Trust Registry Infrastructure group establishes a WE BUILD Digital Directory aligned with the EU Digital Directory standards.
- WP4 QTSP group, in consultation with WP4 Architecture and WP4 Wallet Providers groups, specifies an optional API access protocol as an abstraction layer between the wallet and the QERDS.
- WP4 QTSP group facilitates pilots using (EU) 2025/1944 and eDelivery AS4.
- WP4 Architecture and WP4 QTSP groups jointly evaluate the requirements for a potential "AS5" alternative QERDS protocol, based on the needs of WP2/WP3 use cases.

If use cases require gateways to the designated QERDS, in principle the use cases are responsible for providing those gateways.

Consequences

Testing and piloting with a designated QERDS makes it easier:

- To implement use cases for the EUBW that requires automation and/or agent-based access.
- To connect with organisations responsible for authentic sources for the retrieval and/or verification of attributes, which is necessary for the issuance of qualified electronic attestations of attributes (see Feature: Verification of attributes).
- To develop the QERDS aspect of the envisioned EBW ecosystem, building upon existing ecosystems like the Once-Only Technical System and Peppol, using the WE BUILD ecosystem and its Interoperability Test Bed to provide a meaningful and collaborative context.

Open issues:

- Adapting end-to-end encryption in the four-corner model to the European Business Wallet.
- Support hardware bound credentials and differentiated level of assurance.

The WP4 Architecture group can provide the necessary design competence.

The following risks need to be addressed:

- QERDS and eDelivery are new subject matter to several wallet providers and implementers. They may be tempted to instead mold the well-known OpenID4VC protocols to use cases that are better suited for QERDS. To address this risk, WP4 Architecture group members and in particular Use Case Sync Leads are expected to learn about QERDS and engage with the WP2/WP3 groups to learn which practical use cases will drive adoption of automation and agent-based flows that are the main motivators for using QERDS in the EBW.
- The QERDS requires several cross-group implementations. To reduce interoperability risks, the

WP4 QTSP group should:

- specify at least two Conformance Specifications in consultation with stakeholder groups:
  - EBW-to-QERDS;
  - QTSP-to-QTSP for QERDS;
  - (if use cases need it) EUDIW-to-QERDS;
- work with the WP4 Testing group to perform testing using these Conformance Specifications on the Interoperability Test Bed.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision. In the decision making process, we have heard the following advice.

- 2026-02-03, Andrew Freund, D-Trust, Germany: OK, with processed suggestions regarding detailed requirements and the Interoperability Test Bed.
- 2026-02-11, Alejandro Nieto, DigitelTS, Spain: OK, agreed with the proposed standards and protocols.
- 2026-02-11, Rune Kjørlaug, OpenPeppol, Belgium: Contributed comments regarding directory functions (parked for a next ADR), AS4 suitability, protocol migrations and avoiding lock-in, co-existence with existing ecosystems like Peppol.
- 2026-02-12, Giuseppe De Marco, Dipartimento per la trasformazione digitale, Italy: Recommendation to consider relying party intermediary architectures when designing QERDS gateways.

Attestation Revocation Mechanism

Authors:

- Artur Reaboi, e-Governance Agency, Republic of Moldova
- Alexandru Cozlovschi, e-Governance Agency, Republic of Moldova
- Fabrizio Notarnicola, InfoCamere, Italy
- Alessandro Bazzolo, InfoCamere, Italy

Context

Revocation is the process by which an attestation (including PID/EBWOID) is invalidated before its natural expiry so that it can no longer be trusted or used. While short-lived attestations (expiring in 24 hours or less) do not require revocation, long-term attestations need a standardized mechanism to handle invalidation due to reasons such as lost devices, data inaccuracy, or regulatory changes.

The architecture must ensure that the revocation status check preserves the privacy of the Wallet Holder (herd privacy) and allows for scalable implementation. Additionally, the solution must align with the OpenID4VC HAIP 1.0 specifications.

Decision

The WE BUILD project adopts the IETF Token Status List as the mechanism for semantics, formats, and protocols regarding revocation of attestations.

Consequences

Easier:

- Alignment with OpenID4VC HAIP 1.0 for SD-JWT format is ensured, facilitating interoperability.
- Relying Parties have a standardized format to check for validity across different issuers.

More Difficult:

- To ensure performance and privacy, Issuers must implement complex state management. One way to do it is to pre-allocate random indices in batches, rather than simple sequential generation.
- As Issuers have the sole right to revoke PIDs/EBWOIDs, Authentic Sources and Issuers must establish protocols to notify the Issuer of events requiring revocation (e.g., data changes or lost devices), as they cannot directly revoke an attestation.

Risks:

- Offline verification scenarios require Relying Parties to cache revocation lists. To address this, Issuers should include expiration dates and time-to-live (TTL) in revocation info to drive caching decisions.

Impact (resulting from ADR High-Level Requirements):

- **Issuers** (PID/EBWOID Providers, EAA Providers, including QEAA, Pub-EAA) MUST implement attestation revocation for applicable attestations and that are valid for more than 24h. Revocable attestations MUST be assigned a random status list and random index within it before being signed, all in batches.
- **Issuers** MUST ensure the invalid status for not yet expired and revoked attestations is published within a reasonable amount of time (for instance, under 1h).
- **Wallet Providers** SHOULD ensure their Wallet Units regularly check the revocation status of its PIDs/EBWOIDs and other attestations, and notify the User if any is revoked.
- **Relying Parties** SHOULD check the revocation status via a Revocation Status Service. If reliable information is unavailable, they SHOULD perform a risk analysis rather than a mandatory failure.

Advice

Once merged, this is our consortium's decision. This does not mean all participants agree it is the best possible decision.

In the decision making process, we have heard the following advice:

- 2026-02-04, Jonas Toeniss, Brønnøysundregistrene (BRC), Norway: Move Impact items to

Consequences section

- 2026-02-13, Feedback in WP4 Architecture meeting: Clarify that regular check of WUA revocation is a MUST only for PID/EBWOID Providers
- 2026-02-17, Eelco Klaver, Credenco B.V., Netherlands: WUA implementation and impact to be decided in a separate ADR

Appendix H. Conformance Specifications (CS)

About

The Conformance Specifications define how WE BUILD participants implement wallet interfaces and communication protocols between issuers, wallets, and relying parties (RPs). They ensure interoperability and conformance by translating ADR decisions into precise implementation requirements.

The ITB will be based on the CS as a starting point. The test suites in the ITB are relying predominantly on the CS.

Contributing

The Architecture group define the CS, with help from all implementing participants (wallets, issuers and verifiers). For members of the consortium, find more information on the CS processes are held on open social - Architecture Group.

CS Process Summary for WE BUILD Large Scale Pilots (LSPs)

Conformance Specifications (CS) progress through the following process towards the Large Scale Pilots (LSPs):

[CS Process and Roles] | *https://github.com/webuild-consortium/wp4-architecture/blob/main/images/WPRoles.png*

Approved CSs

CS #	CS Title
CS-001	Credential Issuance - v1.0
CS-002	Credential Presentation - v1.0

CSs Under Development

CS #	CS Title

Conformance Specifications

WE BUILD - Conformance Specification: <TITLE>

Version <VERSION>

Table of Contents

1. Introduction

This document defines the **WE BUILD Conformance Specification for <TITLE>**.

Its purpose is to describe how relevant actors within the WE BUILD ecosystem are expected to interoperate in a consistent and testable way.

This specification should:

- identify the relevant protocol or functional area
- clarify which actors are involved
- define the main requirements needed for interoperability
- support implementation and conformance testing

This specification is based on <BASE STANDARD / PROFILE / ADR> and should be read together with other applicable WE BUILD specifications where relevant.

2. Scope

This specification defines the conformance expectations for <DOMAIN / CAPABILITY>.

It should make clear:

- what this specification covers
- which roles are in scope
- which capabilities are required
- what is out of scope

Out-of-scope items should be listed where needed, especially if they are handled by other WE BUILD documents or external specifications.

3. Normative Language

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** are to be interpreted as described in RFC 2119.

These keywords indicate the strength of requirements for conforming implementations.

4. Roles and Components

This section identifies the roles and components relevant to this specification.

Only roles that matter for this specification should be included.

Examples may include:

- **Wallet Unit (WU):** software acting on behalf of the Holder
- **Holder:** person or organisation controlling the Wallet Unit
- **Issuer:** entity issuing credentials or attestations
- **Verifier:** entity requesting and validating presentations
- **Authorisation Server:** component supporting OAuth / OpenID interactions
- **Trust Provider:** component or service publishing trust-related information

5. Protocol Overview

This section gives a short explanation of how the protocol or function works at a high level.

It should help the reader understand:

- which standards or mechanisms are used
- how the main actors interact
- which security or trust features are important
- what the overall outcome of the interaction is

This section should stay concise. Detailed behaviour belongs in later sections.

6. High-level Flows

This section describes the main interaction flows between actors.

Flows should be written as step-by-step sequences that help implementers understand how the protocol operates.

Example subsections:

6.1 <Flow Name>

Describe:

- participating actors
- how the interaction begins
- the main sequence of actions
- the expected outcome

Example:

1. <STEP 1>
2. <STEP 2>
3. <STEP 3>

7. Normative Requirements

This section defines the normative requirements for implementations.

Requirements may be grouped in the way that best fits the specification, for example by:

- role
- component
- capability
- protocol step

Example structure:

7.1 <Role or Component>

<ROLE OR COMPONENT> **MUST**:

1. <REQUIREMENT 1>
2. <REQUIREMENT 2>

<ROLE OR COMPONENT> **SHOULD**:

1. <RECOMMENDATION 1>

<ROLE OR COMPONENT> **MUST NOT**:

1. <PROHIBITED BEHAVIOUR>

8. Interface Definitions

This section describes the technical interfaces used by the protocol.

Examples may include:

- HTTP endpoints
- wallet invocation URLs
- metadata endpoints
- credential request structures
- presentation responses
- trust registry queries

For each interface, describe:

- direction of communication
- transport method
- request parameters
- response structure

Example subsection:

8.1 <**Interface Name**>

Direction: <SENDER> → <RECEIVER>
Method: <HTTP METHOD>

Request

- <FIELD 1>
- <FIELD 2>

Response

- <FIELD 1>
- <FIELD 2>

Example (illustrative only):

```
&lt;EXAMPLE REQUEST OR URL&gt;
```

9. Conformance

An implementation **conforms to this specification** if it implements the requirements defined in this document for its role and supports the relevant interfaces and flows.

Where relevant, conformance may be stated separately for each role.

Example

An implementation conforms as a **<ROLE 1 CONFORMANCE CLASS>** if it:

1. implements the applicable requirements in Section 7
2. supports the relevant interfaces and flows in Sections 6 and 8
3. supports any required standards or formats referenced by this specification

Additional WE BUILD profiles may define stricter requirements for specific use cases. Such profiles **MUST NOT** weaken the mandatory requirements in this specification.

References

[1] <ORGANISATION> (<YEAR>) <TITLE>. Available at: <URL> (Accessed: <DATE>).

[2] <ORGANISATION> (<YEAR>) <TITLE>. Available at: <URL> (Accessed: <DATE>).

[3] <ORGANISATION> (<YEAR>) <TITLE>. Available at: <URL> (Accessed: <DATE>).

WE BUILD - Conformance Specification: Credential Issuance

Version 1.0

Table Of Contents

1. Introduction

This document defines the **WE BUILD Consortium Conformance Specification (CS)** for high assurance credential issuance based on the decision recorded in WE BUILD ADR Base Protocols.

It profiles:

- OpenID for Verifiable Credential Issuance (OpenID4VCI) v1.0 [1]
- The OpenID4VC High Assurance Interoperability Profile (HAIP) 1.0 - Implementers Draft 1 [2]

The aim is to ensure that Wallet Units and Credential Issuers within the WE BUILD ecosystem interoperate consistently for the **issuance of SD-JWT-VC credentials** [3] with high security and

privacy.

This specification focuses **only on issuance**. Presentation, verification of requirements, and trust management are out of scope and covered in separate documents. The document is used to build the WE BUILD Interoperability Test Bed Plus (ITB+) [4].

2. Scope

This specification defines:

- A profile of OpenID4VCI for issuing SD-JWT-VC credentials
- Requirements for:
  - Wallets that receive credentials
  - Credential Issuers and their Authorisation Servers
- Support for:
  - Wallet-initiated issuance
  - Issuer-initiated issuance via Credential Offer

This document describes:

- Protocol flows for high assurance issuance
- Interfaces and endpoints, including Wallet invocation, Credential Offer, Pushed Authorisation Requests (PAR), Token Endpoint, Credential Endpoint and metadata

3. Normative Language

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL are to be interpreted as commonly used in technical specifications.

4. Roles and Components

This specification uses the following roles:

- **Wallet Unit (WU):** A client application or component acting on behalf of the Holder to obtain and store Verifiable Credentials.
- **Holder:** The subject or representative of the subject who controls the Wallet Unit.
- **Attestation Provider (Issuer):** The entity that decides to issue Verifiable Credentials and controls issuance policy.
- **Authorisation Server (AS):** The OAuth 2.0 and OpenID provider responsible for authenticating the user and issuing tokens for the Issuer. It may be co-located with the Issuer.

5. Protocol Overview

The WE BUILD issuance profile is based on the OAuth 2.0 Authorisation Code Flow with the following mandatory features:

- Authorisation Code and Pre-Authorised Code Flow for all issuance interactions
- OpenID4VCI SD-JWT-VC credential format profile (See NOTE **CS01_01**)
- Sender-constrained tokens, for example, using Demonstration of Proof of Possession (DPoP) or mutual TLS
- PKCE with `S256` code challenge method
- Pushed Authorisation Requests (PAR) for all authorisation requests
- *Wallet Unit Attestation (WUA) for client authentication as defined in OpenID4VCI-based ADR XX (To be written).*

Issuance can be:

- **Wallet-initiated**: the Holder starts from the WU and selects a credential type
- **Issuer-initiated**: the Issuer provides a **Credential Offer** that the WU consumes

Both modes are required in this profile.

> [!NOTE] CS01_01: ISO18013-5 and ISO18013-7 will be supported in subsequent versions, based on use-case requirements.

6. High-level Flows

This section presents the flows as text-based sequence descriptions.

6.1 Wallet-initiated Issuance Flow

Actors: Holder, WU, Issuer (AS and Credential Issuer).

6.1.1 Configuration and discovery

1. The WU retrieves Issuer metadata, which includes:
    - OAuth and OpenID configuration
    - Credential Issuer metadata
    - Mapping between credential types and `scope` values

6.1.2 User selects credential

1. The Holder chooses a credential type, for example, a PID, QEAA or business credential.
2. The WU selects the appropriate Issuer and the corresponding `scope` value.

6.1.3 Pushed Authorisation Request (PAR)

1. The WU constructs an authorisation request containing, at a minimum:
    - `client_id`
    - `scope` identifying the credential type
    - `code_challenge` using PKCE `S256`

- `redirect_uri`
- `response_type=code`
- `state`
- `nonce`

2. The WU sends this request to the Issuer's PAR endpoint, with client authentication bound to the WUA. \
3. The PAR endpoint returns a `request_uri` and a validity period.

6.1.4 User authorisation

1. The WU directs the Holder's user-agent to the Authorisation Endpoint with the `request_uri` obtained from PAR.
2. The Holder authenticates to the AS in accordance with the Issuer's policy.
3. The Holder consents to the issuance of the requested credential.
4. The AS redirects back to the WU with an authorisation `code` and `state`.

6.1.5 Token request

1. The WU sends a token request to the Token Endpoint, including:
   - `grant_type=authorization_code`
   - `code`
   - `redirect_uri`
   - `code_verifier` matching the earlier `code_challenge`
   - client authentication using WUA
2. The Token Endpoint validates the request and returns:
   - sender-constrained `access_token`
   - optional `refresh_token` for credential refresh

6.1.6 Credential request

1. The WU sends a request to the Credential Endpoint containing:
   - `Authorization: Bearer {access_token}`
   - the requested credential format (SD-JWT-VC / mdoc)
   - a `proof` object using the `JWT` proof type that binds the credential to the WU's subject key
2. The Credential Issuer validates:
   - the access token and its sender-constraining mechanism
   - the proof JWT
   - issuance policy
3. The Issuer returns the issued SD-JWT-VC.

6.1.7 Storage

1. The WU validates the credential signature and Issuer binding.
2. The WU stores the credential under the Holder's control.

6.2 Issuer-initiated Issuance via Credential Offer

Actors: Holder, WU, Issuer.

6.2.1 Issuance decision

1. The Holder interacts with the Issuer, for example, digital onboarding, customer due diligence or contract signing.
2. Following successful internal checks, the Issuer decides to issue one or more credentials.

6.2.2 Credential Offer creation

1. The Issuer constructs a **Credential Offer object** that includes:
    - the `credential_issuer` identifier
    - grant information for the `authorization_code` grant type
    - one or more identifiers for supported credential types
    - for each type, a `scope` value that maps unambiguously to that credential type

6.2.3 Credential Offer delivery and Wallet invocation

1. The Issuer delivers the offer to the Holder by one of:
    - displaying a QR code that encodes a URL which uses the `openid-credential-offer://` scheme to invoke the WU
        - sending a link that uses the `openid-credential-offer://` scheme to a device with a registered WU
2. Both same-device and cross-device delivery methods MUST be supported.

6.2.4 WU processes the offer

1. The WU is invoked via `openid-credential-offer://` and receives the Credential Offer.
2. The WU parses the offer and determines:
    - Issuer base URL
    - offered credential types
    - associated `scope` values used for authorisation
3. The WU displays the offer to the Holder and asks for confirmation to proceed.

6.2.5 Authorisation and token exchange

1. The WU initiates the Authorisation Code Flow using PAR as defined in Section 6.1, reusing `scope` values from the offer.

2. The remainder of the flow, including authorisation, token request, credential request and storage, is identical to the Wallet-initiated flow.

6.2.6 Credential Request

The Wallet sends a Credential Request (or Batch Request) to the Credential Endpoint, including:

- `Authorization: Bearer {access_token}`
- SD-JWT-VC configuration
- `proof` object

6.2.7 Deferred Credential Request

If issuance cannot be completed immediately, the Issuer returns:

- `acceptance_token`
- optional `interval` (retry hint)

Batch requests may contain both immediate and deferred items.

6.3 Deferred Credential Request

Deferred issuance applies to both wallet-initiated and issuer-initiated flows.

When the Credential Issuer cannot immediately produce one or more credentials:

1. The Issuer returns:
   - `acceptance_token`
   - optional `interval` (retry hint) \
2. The WU MUST store the acceptance_token associated with the pending credential(s). \
3. The WU periodically retries using the acceptance_token until:
   - the credential is successfully issued, or
   - The Issuer signals an unrecoverable error. \
4. Batch requests may contain a mix of immediate and deferred items. Each deferred item receives its own acceptance_token and can be polled independently.

7. Normative Requirements

This section summarises the mandatory requirements for WE BUILD implementations.

7.1 Common requirements (WU and Issuer)

Both WU and Issuer **MUST**:

1. Support the Authorisation Code Flow as the only flow for credential issuance.
2. Support the SD-JWT-VC credential format profile as defined for OpenID4VCI.

3. Support sender-constrained tokens, for example, using DPoP or mutual TLS.
4. Support PKCE with the `S256` code challenge method for all authorisation requests.
5. Support Wallet-initiated and Issuer-initiated issuance.

7.2 Credential Offer

Issuers **MUST**:

1. Support the grant type `authorization_code` in Credential Offers, aligned with OpenID4VCI.
2. Include a `scope` value for each offered credential type so that the Wallet can identify the correct type and use the same value in the authorisation request.
3. Support both same-device and cross-device sending of Credential Offers.
4. Support at least the `openid-credential-offer://` custom URL scheme for Wallet invocation.

WUs **MUST**:

1. Be able to parse a Credential Offer that uses `authorization_code` as the grant type.
2. Use the `scope` value from the offer in the authorisation request.
3. Support invocation via the `openid-credential-offer://` custom URL scheme.

7.3 Authorisation Endpoint and PAR

Issuers **MUST**:

1. Require Pushed Authorisation Requests (PAR) for all authorisation requests. Direct front-channel authorisation requests without PAR MUST NOT be used.
2. Ensure that the Wallet authenticates at the PAR endpoint using the same method as used for client authentication at the Token Endpoint.

WUs **MUST**:

1. Use PAR for all authorisation requests.
2. Use the `scope` parameter to indicate the credential type to be issued. Each `scope` value MUST map to a specific credential type that is known from Issuer metadata or from the Credential Offer.
3. Ensure that the `client_id` in the PAR request matches the `sub` claim in the Wallet attestation JWT used for client authentication.

7.4 Token Endpoint and Wallet Attestation

WUs **MUST**:

1. Perform client authentication at the Token Endpoint using wallet attestation as defined in Annexe E of the OpenID4VCI specification.
2. Include the public key, and optionally a trust chain, used to validate the Wallet attestation in the `x5c` JOSE header of the attestation JWT.
3. Ensure the `sub` claim in the Wallet attestation JWT equals the `client_id` used in PAR and token

requests.

Issuers **SHOULD**:

1. Support refresh tokens for credential refresh, following OpenID4VCI guidance on refresh usage and lifetime.

7.5 Credential Endpoint

Issuers **MUST**:

1. Support the `JWT` proof type in the Credential Endpoint.
2. Support the SD-JWT-VC credential format and validate the proof binding between the Wallet subject and credential.

Wallets **MUST**:

1. Send a proof JWT that contains claims required by the Issuer to bind the credential to the Wallet's subject key.
2. Validate the returned SD-JWT-VC, including:
    - signature
    - Issuer identifier
    - key binding and any status information, according to the SD-JWT-VC profile

7.6 Deferred Credential Endpoint

Issuers **MUST**:

- Support a `deferred_credential_endpoint`.
- Return `acceptance_token` when issuance is delayed.
- Validate `acceptance_token` and ensure proper lifetime and binding.
- Publish endpoint in metadata.

Issuers **SHOULD**:

- Provide clear retry guidance.
- Return explicit errors when expired or failed.

Wallets **MUST**:

- Recognise deferred responses and store the `acceptance_token`.
- Call the Deferred Credential Endpoint until the credential is ready or the transaction ends.
- Distinguish *pending* vs *failed* issuance in UI.

Wallets **SHOULD**:

- Apply poll intervals/back-off.

- Allow users to stop polling.

7.7 Server Metadata

Issuers **MUST** publish metadata that includes:

1. OAuth 2.0 and OpenID configuration, including Authorisation, Token and PAR endpoints.
2. Credential Issuer metadata that describes:
    - all supported credential types
    - a mapping from each credential type to a unique `scope` value

Wallets **MUST**:

1. Retrieve and process Issuer metadata, including the mapping from credential type to `scope`.
2. Use this mapping when constructing authorisation requests and when interpreting Credential Offers.

8. Interface Definitions

This section defines the logical interfaces for conformance. Exact URL paths are deployment-specific and discovered through metadata.

8.1 WU Invocation Interface

- **Direction**: Issuer to Wallet
- **Transport**: custom URL scheme and optional QR code
- **Requirement**:
    - Wallets and Issuers MUST support the `openid-credential-offer://` scheme as a minimal mechanism to invoke Wallets in both same-device and cross-device scenarios

Example (illustrative)

```
openid-credential-offer://credential-offer?request_uri=...
```

The concrete parameters and encoding follow HAIP and OpenID4VCI guidance on Credential Offers.

8.2 Credential Offer Interface

- **Direction**: Issuer to WU

The **Credential Offer object** MUST contain at least:

- `credential_issuer`: base URL identifying the Issuer
- `grants`: object that includes support for `authorization_code`
- For each credential type:
    - a credential type identifier

    - the associated `scope` value

The exact JSON structure MUST comply with OpenID4VCI Credential Offer definitions.

8.3 PAR Endpoint

- **Direction**: Wallet to Issuer (AS)
- **Method**: `POST`

Request (logical fields)

- `client_id`
- `scope`
- `code_challenge` using PKCE `S256`
- `code_challenge_method=S256`
- `redirect_uri`
- `response_type=code`
- `state`, `nonce`

Response

- `request_uri`
- `expires_in`

All PAR requests MUST be client-authenticated according to Section 7.4.

8.4 Token Endpoint

- **Direction**: WU to Issuer (AS)
- **Method**: `POST`

Request (logical fields)

- `grant_type=authorization_code`
- `code`
- `redirect_uri`
- `code_verifier`
- client authentication using Wallet attestation JWT, for example, `client_assertion` and `client_assertion_type`

Response

- `access_token` (sender-constrained)
- `token_type`
- `expires_in`

- optional `refresh_token`

8.5 Credential Endpoint

- **Direction**: WU to Issuer
- **Method**: `POST` **Request (logical fields)**
- HTTP header: `Authorization: Bearer {access_token}`
- Body:
  - `format` (for example, `vc+sd-jwt` or the identifier used in the chosen SD-JWT-VC profile)
  - identification of the requested credential configuration
  - `proof` object with:
    - `proof_type="jwt"`
    - `jwt` containing proof claims

Response

- SD-JWT-VC credential and any associated metadata defined by the OpenID4VCI SD-JWT-VC profile

8.7 Deferred Credential Endpoint

Direction: WU → Issuer
Method: POST **Request (logical fields)**

- HTTP header:
  - `Authorization: Bearer {token}`
  - `Content-Type: application/json`
- Body parameters:
- `transaction_id`

Response

A Deferred Credential Response MAY either provide the issued credentials or indicate that issuance is still pending.

If credential issuance is complete:

- The response MUST contain the **credentials** parameter
- HTTP status code MUST be **200 (OK)**.

If credential issuance is still pending

- The response MUST contain:
  - **transaction_id**: MUST match the request.
  - **interval:** recommended waiting time before retrying.

- HTTP status code MUST be **202 (Accepted)**.

Error Response If the Deferred Credential Request is invalid, the Issuer returns an error response.

- `invalid_transaction_id:` Indicates that the `transaction_id` was not issued by the Credential Issuer or has already been used.
- If the Credential Issuer can no longer issue the credential(s), it returns `credential_request_denied`. The WU stops retrying for the given `transaction_id`.

8.8 Metadata Endpoints

Issuers **MUST** publish:

- OpenID Provider and OAuth discovery document
- Credential Issuer metadata document

The latter MUST include:

- supported credential types
- for each type, the associated `scope` value

WU uses these documents for dynamic configuration.

9. Conformance

An implementation **conforms to this specification as a Wallet Provider** if it:

1. Implements the WU requirements in Sections 6 and 7.
2. Supports the interfaces defined for WU behaviour in Section 8.
3. Uses SD-JWT-VC and OpenID4VCI as profiled by the OpenID4VC High Assurance Interoperability Profile Implementer's Draft, Section 4.

An implementation **conforms to this specification as an Issuer** if it:

1. Implements the Issuer requirements in Sections 6 and 7.
2. Publishes server metadata, including type to `scope` mappings.
3. Provides the PAR, Token, Credential and WU invocation interfaces described in Section 8.

Profiles may define additional constraints for specific WE BUILD credential types, such as PID, QEAA, or business credentials. Such profiles MUST NOT relax the mandatory requirements in this document. The specific issuance will be taken into a separate CS.

References

[1] OpenID Foundation (2025) OpenID for Verifiable Presentations 1.0. OpenID Foundation. Available at: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html (Accessed: 24 November 2025).

[2] OpenID Foundation (2025) OpenID4VC High Assurance Interoperability Profile — draft 03. OpenID Foundation. Available at: https://openid.net/specs/openid4vc-high-assurance-interoperability-profile-1_0-ID1.html (Accessed: 24 November 2025)

[3] IETF (2025) SD-JWT-based Verifiable Credentials. IETF. Available at: https://www.ietf.org/archive/id/draft-ietf-oauth-sd-jwt-vc-09.html (Accessed: 24 November 2025).

[4] WE BUILD (2025) Interoperability Test Bed - Reference Specification, 12 November, Available at: https://github.com/webuild-consortium/wp4-interop-test-bed/blob/main/docs/reference-implementation-interoperability-test-bed.md (Accessed: 24 November 2025).

WE BUILD - Conformance Specification: Credential Presentation

Version 1.0

Table Of Contents

1. Introduction

This document defines the **WE BUILD Conformance Specification for Credential Presentation**, describing how Wallet Units (WU) and Verifiers interoperate using OpenID for Verifiable Presentations (OpenID4VP) 1.0 [1] in alignment with the OpenID4VC High Assurance Interoperability Profile (HAIP) 1.0 - Implementer's Draft 1 [2], based on the decision recorded in WE BUILD ADR Base Protocols.

It specifies a high-assurance presentation profile for use within the WE BUILD ecosystem, covering:

- Presentation request and response flows
- Interfaces between Wallets and Verifiers
- Security, privacy and interoperability requirements
- Support for SD-JWT-VC credentials [3]
- Same-device and cross-device invocation patterns

This document complements the WE BUILD Conformance Specification: Credential Issuance v1.0. The document is used to build the WE BUILD Interoperability Test Bed Plus (ITB+) [4].

2. Scope

This specification defines the conformance profile for high-assurance credential presentation:

- Requirements for:
  - WUs that respond to presentation requests
  - Verifiers that initiate presentation requests
- Mandatory features:
  - OpenID4VP 1.0
  - HAIP ID-1 Section 5 requirements
  - JWT-based Presentation Proof

- SD-JWT-VC selective disclosure
- Same-device and cross-device invocation
- openid4vp:// Wallet invocation

3. Normative Language

The terms MUST, MUST NOT, SHOULD, SHOULD NOT, REQUIRED, RECOMMENDED, MAY and OPTIONAL are to be interpreted as described in RFC 2119.

4. Roles and Components

This specification uses the following roles:

- **Wallet Unit (WU):** A client application or component acting on behalf of the Holder to obtain and store Verifiable Credentials.
- **Holder:** The subject or representative of the subject who controls the Wallet Unit.
- **Verifier:** Entity requesting verifiable presentations, validating responses and making authorisation decisions.

5. Protocol Overview

The WE BUILD presentation profile is based on OpenID4VP with the following mandatory features defined by HAIP ID-1:

- JWT-Secured Authorisation Request (JAR): All authorisation requests MUST be signed.
- Digital Credentials Query Language (DCQL): MUST be used for querying credentials.
- Client Identifier Schemes: Usage of `x509_san_dns` or `verifier_attestation`, `decentralized_identifiers` is also recommended (`did:web`, `did:jwk`)
- Crypto Suites: Strict adherence to P-256 (secp256r1) with ES256 for signing.
- Holder Binding: Mandatory Key Binding JWT (KB-JWT) for SD-JWT VCs. (See **NOTE_CS02_01**)

High-level steps:

1. Verifier creates Presentation Request
2. Wallet is invoked via openid4vp:// (same or cross device)
3. Wallet validates Presentation Request
4. Holder consents
5. Wallet generates Presentation Proof + Disclosures
6. Wallet submits Presentation Response
7. Verifier validates and produces outcome

NOTE_CS02_01: ISO18013-5 and ISO18013-7 will be supported in subsequent versions based on use case requirements.

6. High-level Flows

This chapter defines the presentation flows required by WE BUILD.

6.1 Same-device Presentation Flow

6.1.1 Presentation Request Creation

The Verifier prepares a signed Presentation Request Object containing:

- Requested credential types
- Disclosure constraints
- Proof requirements (nonce, audience)
- Expiry (exp)
- Verifier identifier (client_id)

The request MUST be integrity-protected (JAR-style or equivalent).

6.1.2 WU Invocation

The Verifier redirects the user-agent to the WU using:

```
openid4vp://present?request_uri=<URL>
```

Wallet retrieves or validates the signed Presentation Request Object.

6.1.3 WU Validation

The WU MUST validate:

- Signature of Presentation Request
- Nonce freshness
- Audience matches Wallet
- Expiry validity
- Credential types and disclosure constraints
- Request integrity

Unsigned or invalid requests MUST be rejected.

6.1.4 Holder Consent

The WU MUST display:

- Verifier identity
- Requested credential types
- Requested attributes or claims

- Any selective disclosure details

Holder MUST explicitly consent.

6.1.5 Presentation Generation

Upon consent, the Wallet MUST generate:

- JWT-based Presentation Proof
- Selective disclosures for SD-JWT-VC
- Binding between:
  - Presentation Proof and Wallet-held key
  - Nonce
  - Audience

6.1.6 Presentation Submission

The Wallet MUST POST the Presentation Response to the Verifier's Presentation Endpoint, including:

- `vp_token` containing the JWT-encoded Presentation
- format specifying SD-JWT-VC

Sender-constrained token usage MUST be applied if configured.

6.1.7 Result Handling

The Verifier MUST return:

- A success object if verification succeeded
- Error information when the presentation is invalid or incomplete

Wallet MUST correctly display the outcome to the Holder.

6.2 Cross-device Presentation Flow

6.2.1 Presentation Request Creation and Display

Verifier constructs the Presentation Request Object (as in 6.1.1) and encodes it in a QR-based `openid4vp://` URL.

6.2.2 Wallet Unit Invocation via QR

Holder scans the QR code. WU retrieves the Presentation Request Object (embedded or via `request_uri`).

6.2.3 Wallet Validation

The same validation rules as 6.1.3 apply.

6.2.4 Holder Consent

Same as 6.1.4.

6.2.5 Presentation Generation

Same as 6.1.5.

6.2.6 Presentation Submission

WU delivers the Presentation directly to the Verifier's Presentation Endpoint (back channel). Redirection flow MAY be used if supported.

6.2.7 Result Handling

Verifier processes the Presentation and returns the outcome as in 6.1.7.

7. Normative Requirements

7.1 Wallet Unit Requirements

Wallets MUST:

1. Support HAIP-compliant OpenID4VP.
2. Support the same-device and cross-device flows.
3. Support openid4vp://present invocation.
4. Validate signed Presentation Requests.
5. Implement SD-JWT-VC selective disclosure.
6. Provide transparent Holder consent.
7. Generate JWT-based Presentation Proof.
8. Bind Presentation Proof to Verifier's nonce and audience.
9. Submit Presentation Responses to the Presentation Endpoint.

Wallets MUST NOT:

- Accept unsigned or invalid Presentation Requests
- Auto-consent
- Add unsolicited claims

7.2 Verifier Requirements

Verifiers MUST:

1. Create a signed Presentation Request Object.
2. Use nonces and audience restrictions.
3. Support same-device and cross-device invocation.

4. Publish Verifier Metadata.
5. Provide a Presentation Endpoint.
6. Validate all Presentation Responses, including:
    - Signature of Presentation Proof
    - Credential authenticity
    - SD-JWT-VC disclosure integrity
    - Holder binding
    - Nonce and audience binding
    - Satisfaction of request constraints

Verifiers MUST NOT:

- Request unnecessary personal information
- Disable nonce or audience validation

8. Interface Definitions

Interfaces in this chapter follow the structure from the Issuance Conformance Specification.

8.1 Wallet Invocation Interface

Direction: Verifier → Wallet
Transport: `openid4vp://` scheme
Usage: Same-device or cross-device scanning

Example:

```
openid4vp://present?request_uri=https://verifier.example.org/request/123
```

Wallet MUST retrieve or validate the Presentation Request Object.

8.2 Presentation Request Object Interface

The Presentation Request Object MUST include:

- Verifier identifier (`client_id`)
- `nonce`
- `audience`
- Requested credential types
- Disclosure constraints
- Proof requirements
- Expiry

- Signature (integrity-protected object)

Wallet Units reject incomplete or invalid request objects.

8.3 Presentation Endpoint

Direction: Wallet → Verifier
Method: POST
Authentication: MAY use sender-constrained tokens

Request Body Example

```
{ "vp_token": "<JWT-Presentation>", "format": "vc+sd-jwt" }
```

Success Response Example

```
{ "status": "ok" }
```

Error Example

```
{ "error": "invalid_presentation", "error_description": "Nonce invalid or expired" }
```

8.4 Verifier Metadata Interface

Verifiers MUST publish metadata containing:

- Presentation_endpoint
- Supported vp_formats
- Supported proof mechanisms
- JWK set for Request signing
- Required credential types

Wallet Units retrieves this metadata where available.

9. Conformance

An implementation **conforms to this specification as a Wallet Provider** if it:

1. Implements all Wallet requirements in Section 7.1
2. Implements all interfaces and behaviours in Section 8
3. Supports flows defined in Section 6
4. Supports SD-JWT-VC as defined for OpenID4VP

An implementation **conforms to this specification as an Issuer** if it:

1. Implements all Verifier requirements in Section 7.2
2. Publishes required Verifier Metadata
3. Implements the Presentation Request and Presentation Endpoint interfaces
4. Supports both same-device and cross-device flows

References

[1] OpenID Foundation (2025). OpenID for Verifiable Credential Issuance 1.0. OpenID Foundation, 16 September. Available at: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html (Accessed: 24 November 2025).

[2] OpenID Foundation (2025) OpenID4VC High Assurance Interoperability Profile — draft 03. OpenID Foundation. Available at: https://openid.net/specs/openid4vc-high-assurance-interoperability-profile-1_0-ID1.html (Accessed: 24 November 2025)

[3] IETF (2025) SD-JWT-based Verifiable Credentials. IETF. Available at: https://www.ietf.org/archive/id/draft-ietf-oauth-sd-jwt-vc-09.html (Accessed: 24 November 2025).

[4] WE BUILD (2025) Interoperability Test Bed - Reference Specification, 12 November, Available at: https://github.com/webuild-consortium/wp4-interop-test-bed/blob/main/docs/reference-implementation-interoperability-test-bed.md (Accessed: 24 November 2025).

WE BUILD - Conformance Specification: Remote Qualified Signing with Wallet Units

Version 1.0 / Approved Date: 01 April 2026

Authors / Contributors: WP4 Architecture

- Lal Chandran, iGrant.io, Sweden
- Hristian Daskalov, EvroTrust, Bulgaria
- George J Padayatti, iGrant.io, Sweden
- Andreas Abraham, ValidatedID, Spain
- Alejandro Nieto, DigitelTS, Spain
- Hidde Dorhout, Cleverbase, Netherlands
- Andrew Freund, D-Trust, Germany

Table of Contents

1. Introduction

This document defines the **WE BUILD Conformance Specification: Remote Qualified Signing with Wallet Units**, describing how Wallet Units (WU) and Relying Parties interoperate to create qualified electronic signatures using OpenID for Verifiable Presentations (OpenID4VP) 1.0 [1] and the CSC Data Model Bindings [3].

This specification extends the WE BUILD Conformance Specification: Credential Presentation [5] (hereafter CS-02). All requirements defined in CS-02 apply unless explicitly superseded here. This document defines only the signing-specific additions.

> **NOTE_CSRS_00** In this specification, "remote" refers to the interaction pattern between the Signer and the Relying Party. The signing request and response are exchanged over the network via OpenID4VP. It does not imply a remote signing service (QTSP) hosting the signing key server-side. In the wallet-centric model defined here, the WU holds the signing key and generates the signature locally.

It covers:

- Signing request and response flows using the CSC `qesRequest` transaction data type
- Support for the CSC X.509 credential format
- Signer consent requirements specific to qualified electronic signatures
- Inline and out-of-band signed document delivery

> **NOTE_CSRS_01** The `qesApproval` flow (provider-centric model using mdoc or SD-JWT VC approval credentials) is out of scope for this version and will be addressed in a subsequent version.

2. Scope

This specification defines the conformance profile for remote qualified electronic signature creation. It applies in addition to CS-02 [5].

Requirements are defined for:

- Wallet Units that respond to signing requests
- Relying Parties that initiate signing requests

Mandatory features beyond CS-02:

- CSC `qesRequest` transaction data type [3]
- CSC X.509 credential format (`https://cloudsignatureconsortium.org/2025/x509`)
- Signer consent rendering requirements
- AdES signature generation
- Inline and out-of-band (`responseURI`) response delivery

Out of scope:

- All requirements already covered by CS-02 [5]
- CSC API endpoints (`signatures/signDoc`, `signatures/signHash`)
- The qesApproval flow (see NOTE_CSRS_01)
- The credential used for signature request confirmation

3. Normative Language

As defined in CS-02 [5] Section 3.

4. Roles and Components

This specification uses the roles defined in CS-02 [5] Section 4, with the following substitutions and additions:

- **Holder** is referred to as **Signer** in this specification.
- **Verifier** is referred to as **Relying Party (RP)** in this specification.
- The Wallet Unit acts as both credential holder and signing application in the wallet-centric model covered by this specification.

5. Protocol Overview

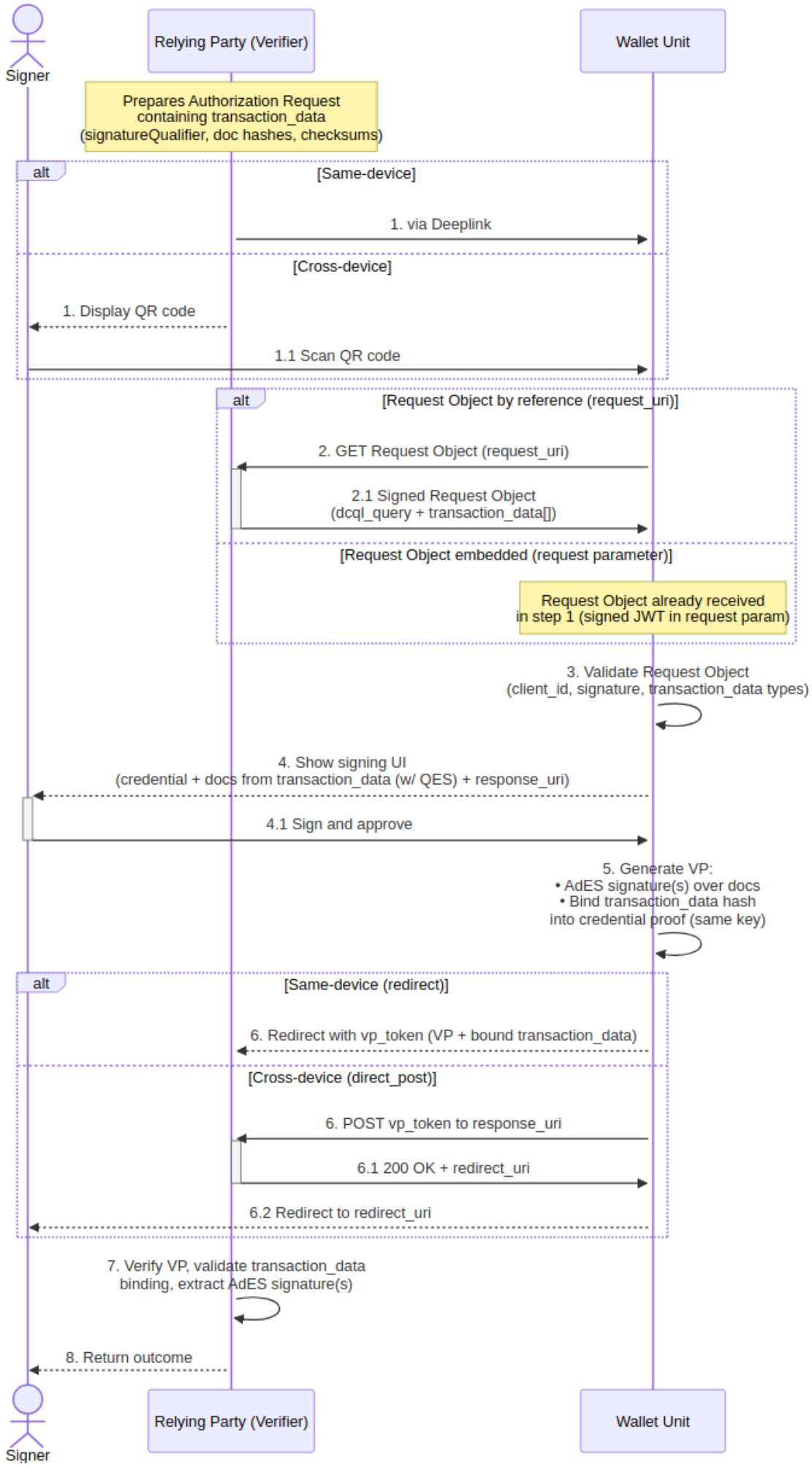
This specification applies the protocol baseline defined in CS-02 [5] Section 5 without modification, with the following signing-specific additions:

- The Authorisation Request MUST include `transaction_data` containing a base64url-encoded `qesRequest` object as defined in CSC-DMB [3] Section 6.2.1
- `signatureQualifier` MUST always be present in the `qesRequest`
- Document integrity MUST be verified by the WU when both `href` and `checksum` are present

High-level steps:

1. Relying Party creates a Signing Request containing `qesRequest` transaction data
2. WU is invoked via `openid4vp://` (same-device) or QR code (cross-device)
3. WU validates the Signing Request
4. Signer reviews documents and consents
5. WU generates AdES signature(s)
6. WU submits the Presentation Response containing the signed document(s)
7. Relying Party validates and returns the outcome

The above flow is illustrated below:


Signer
Relying Party (Verifier)
Wallet Unit
Prepares Authorization Request
containing transaction_data
(signatureQualifier, doc hashes, checksums)
alt
[Same-device]
1. via Deeplink
[Cross-device]
1. Display QR code
1.1 Scan QR code
alt
[Request Object by reference (request_uri)]
2. GET Request Object (request_uri)
2.1 Signed Request Object
(dcql_query + transaction_data[])
[Request Object embedded (request parameter)]
Request Object already received
in step 1 (signed JWT in request param)
3. Validate Request Object
(client_id, signature, transaction_data types)
4. Show signing UI
(credential + docs from transaction_data (w/ QES) + response_uri)
4.1 Sign and approve
5. Generate VP:
• AdES signature(s) over docs
• Bind transaction_data hash
into credential proof (same key)
alt
[Same-device (redirect)]
6. Redirect with vp_token (VP + bound transaction_data)
[Cross-device (direct_post)]
6. POST vp_token to response_uri
6.1 200 OK + redirect_uri
6.2 Redirect to redirect_uri
7. Verify VP, validate transaction_data
binding, extract AdES signature(s)
8. Return outcome
Signer
Relying Party (Verifier)
Wallet Unit

> **NOTE_CSRS_02** ISO18013-5 and ISO18013-7 credential formats will be considered in subsequent versions based on use case requirements.

6. High-level Flows

6.1 Same-device Signing Flow

6.1.1 Signature Request Creation

The Relying Party prepares a signed Presentation Request Object as defined in CS-02 [5] Section 6.1.1, with the following additional requirements:

- The DCQL query MUST contain a credential entry with `format: "https://cloudsignatureconsortium.org/2025/x509"` and optionally `certificatePolicies` or `certificateFingerprints` in the `meta` field
- `transaction_data` MUST contain a base64url-encoded `qesRequest` specifying `type`, `credential_ids`, `signatureQualifier`, and at least one `signatureRequests` entry

6.1.2 WU Invocation

As defined in CS-02 [5] Section 6.1.2.

6.1.3 WU Validation

As defined in CS-02 [5] Section 6.1.3, with the following additional checks:

- `transaction_data` MUST decode to a valid `qesRequest`
- `signatureQualifier` MUST be present and recognised
- The available credential MUST be capable of producing a QES with the specified `signatureQualifier`; if not, the WU MUST abort
- If both `href` and `checksum` are present in a `signatureRequest`, the WU MUST verify document integrity; if verification fails, the WU MUST abort

6.1.4 Signer Consent

In addition to the consent requirements of CS-02 [5] Section 6.1.4, the WU MUST display:

- A clear indication that the transaction creates a qualified electronic signature or seal
- The trust framework identified by `signatureQualifier`
- The label of each document, or a clear indication that no label is provided
- Whether document integrity has been automatically verified
- The URI to which the signed response will be sent, if `responseURI` is specified

The WU SHOULD provide a preview or rendering of the to-be-signed document(s) where technically feasible, to allow the Signer to verify the content before approving.

6.1.5 Signature Generation

Upon consent, the WU MUST generate AdES signature(s) per the `signature_format` and `conformance_level` specified in each `signatureRequest`.

6.1.6 Presentation Submission

As defined in CS-02 [5] Section 6.1.6, with the following additions:

The `vp_token` MUST include a `qes` object containing either:

- `documentWithSignature`: base64-encoded signed document(s), for enveloped formats (e.g. PAdES)
- `signatureObject`: base64-encoded detached signature(s)

If `responseURI` is specified in the `qesRequest`, the WU MUST HTTP POST the `qesResponse` to that URI as defined in Section 8.2, and MUST return an empty credential response in the `vp_token`.

6.1.7 Result Handling

As defined in CS-02 [5] Section 6.1.7.

6.2 Cross-device Signing Flow

The cross-device signing flow follows CS-02 [5] Section 6.2 in its entirety, applying the signing-specific additions defined in Sections 6.1.1 through 6.1.7 of this specification at each corresponding step.

7. Normative Requirements

7.1 Wallet Unit Requirements

In addition to all requirements in CS-02 [5] Section 7.1, Wallet Units MUST:

1. Support the CSC X.509 credential format (`https://cloudsignatureconsortium.org/2025/x509`).
2. Process `qesRequest` transaction data as defined in CSC-DMB [3] Section 6.2.1.
3. Verify document integrity against `checksum` when both `href` and `checksum` are present.
4. Support `data:` URIs with base64 encoding in `href`.
5. Display the signing-specific consent information defined in Section 6.1.4.
6. Generate AdES signatures per the specified `signature_format` and `conformance_level`.
7. Support inline response delivery via `documentWithSignature` or `signatureObject`.
8. Support out-of-band response delivery via `responseURI`.

Wallet Units MUST NOT:

- Proceed if document integrity verification fails.
- Proceed if the credential cannot satisfy the specified `signatureQualifier`.

7.2 Relying Party Requirements

In addition to all requirements in CS-02 [5] Section 7.2, Relying Parties MUST:

1. Include `transaction_data` with a valid `qesRequest` in every signing request.
2. Include `signatureQualifier` in every `qesRequest`.
3. Use the CSC X.509 credential format in the DCQL query.
4. Provide a valid access method (`public` or `OTP`) for all `href` document references.
5. If using `responseURI`: implement an HTTPS endpoint accepting HTTP POST as defined in Section 8.2.

Relying Parties MUST NOT:

- Omit `signatureQualifier` from `qesRequest` objects.

8. Interface Definitions

Interfaces in this specification follow the structure defined in CS-02 [5] Section 8. The Wallet Invocation Interface defined in CS-02 [5] Section 8.1 applies without modification.

8.1 Signing Request Object Interface

The Presentation Request Object MUST satisfy CS-02 [5] Section 8.2, with the following additions.

The DCQL query MUST include a credential entry of the following form:

```
{
  "id": "signing-cert-01",
  "format": "https://cloudsignatureconsortium.org/2025/x509",
  "meta": {
    "certificatePolicies": ["0.4.0.2042.1"]
  }
}
```

The `transaction_data` array MUST contain the following object, base64url-encoded:

```
{
  "type": "https://cloudsignatureconsortium.org/2025/qes",
  "credential_ids": ["signing-cert-01"],
  "signatureQualifier": "eu_eidas_qes",
  "signatureRequests": [
    {
      "label": "Example Contract",
      "access": { "type": "public" },
      "href": "https://rp.example.org/documents/contract.pdf",
      "checksum": "sha256-HZQzZmMAIWekfGH0/ZKW1nsdt0xg3H6bZYztgsMTLw0=",
      "signature_format": "P",
      "conformance_level": "AdES-B-B",
```

```
      "signed_envelope_property": "Certification"
    }
  ]
}
```

> **NOTE_CSRS_03** In this profile, the base64url-decoded `transaction_data` value is the CSC `qesRequest` object with `type` `https://cloudsignatureconsortium.org/2025/qes`. The `credential_ids` values refer to the `id` of the associated DCQL credential query, and not to CSC API `credentialID` values. For interoperability with Wallet Units that validate `transaction_data` strictly, Relying Parties should not include additional profile-specific members unless this specification defines them.

8.2 Presentation Endpoint

The Presentation Endpoint defined in CS-02 [5] Section 8.3 applies. The request body MUST use the following structure for inline signing responses:

```
{
  "signing-cert-01": {
    "qes": {
      "documentWithSignature": ["<base64-encoded signed document>"]
    }
  }
}
```

When `responseURI` *(CSC-DMB parameter, not to be confused with OpenID4VP* `response_uri`*)* is specified in the `qesRequest`, the WU MUST HTTP POST the `qesResponse` to that URI. The following requirements apply:

- `responseURI` MUST use the `https` scheme.
- The WU MUST verify that the `responseURI` host matches the Relying Party's `client_id` or a domain listed in the RP's verified metadata. This prevents the signed output from being redirected to an endpoint not controlled by the authenticated RP, even if the request object itself is properly signed.
- If the HTTP POST to `responseURI` fails (network error or non-2xx response), the WU MUST abort the signing flow and report an error to the Signer.

```
POST <responseURI path> HTTP/1.1
Host: <responseURI host>
Content-Type: application/json

{ "documentWithSignature": ["<base64-encoded signed document>"] }
```

The `responseURI` endpoint MUST return `HTTP 200 OK` on successful receipt.

> **NOTE_CSRS_04** Replay protection for the signing flow is provided by the OpenID4VP `nonce` binding mandated by CS-02, combined with the `transaction_data` hash bound into the VP token. Implementations SHOULD additionally consider including an application-specific nonce in the `qesRequest` as recommended by CSC Data Model Bindings Note 25, to provide defence-in-depth at the CSC layer.

The `vp_token` returned to the Presentation Endpoint MUST be empty:

```
{ "signing-cert-01": {} }
```

8.3 Relying Party Metadata Interface

The Verifier Metadata Interface defined in CS-02 [5] Section 8.4 applies. Relying Parties MUST additionally declare support for the CSC X.509 credential format in `vp_formats_supported`.

9. Conformance

An implementation conforms to this specification as a **Wallet Unit** if it:

1. Conforms to CS-02 [5] Section 9 as a Wallet Provider
2. Implements all Wallet Unit requirements in Section 7.1 of this specification
3. Supports the signing flows defined in Section 6 of this specification

An implementation conforms to this specification as a **Relying Party** if it:

1. Conforms to CS-02 [5] Section 9 as a Verifier (referred to as "Issuer" in CS-02 Section 9)
2. Implements all Relying Party requirements in Section 7.2 of this specification
3. Supports both same-device and cross-device signing flows

References

[1] OpenID Foundation (2025). OpenID for Verifiable Presentations 1.0. Available at: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html (Accessed: 5 March 2026).

[2] OpenID Foundation (2025). OpenID4VC High Assurance Interoperability Profile - Implementer's Draft 1. Available at: https://openid.net/specs/openid4vc-high-assurance-interoperability-profile-1_0-ID1.html (Accessed: 5 March 2026).

[3] Cloud Signature Consortium (2025). CSC Data Model Bindings, version 1.0.0. Published 14 October 2025. Available at: https://cloudsignatureconsortium.org/wp-content/uploads/2025/10/data-model-bindings.pdf (Accessed: 30 March 2026)

[4] Cloud Signature Consortium (2025). Data Model for Remote Signature Applications, version 1.0.0.

Published 16 October 2025. Available at: https://cloudsignatureconsortium.org/wp-content/uploads/2025/10/csc-dm.pdf (Accessed: 30 March 2026)

[5] WE BUILD (2025). Conformance Specification: Credential Presentation, version 1.0. Available at: https://github.com/webuild-consortium/wp4-architecture/blob/main/conformance-specs/cs-02-credential-presentation.md (Accessed: 5 March 2026).

[6] EWC Consortium (2025). RFC-010: Document Signing on a Remote Signing Service Provider using Long-Term Certificates, version 1.1. Available at: https://github.com/EWC-consortium/eudi-wallet-rfcs/blob/main/ewc-rfc010-long-term-certifice-qes-creation.md (Accessed: 5 March 2026).

[7] ETSI EN 319 102-1. Electronic Signatures and Trust Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation.