27-02-2026

Deliverable D1.2:

Final Blueprint Architectures version 2025

Publication Date	2026-02-27
Due Date:	2026-02-28

Authors:	Sander Apweiler (FZJ), Valeria Ardizzone (EGI), Konstantinos Georgilakis (GRNET), Peter Gietz (DAASI), Diana Gudu (KIT), Marcus Hardt (KIT), David Hübner (DAASI), Jens Jensen (STFC UKRI), Ivan Kanakarakis (SUNET), Christos Kanellopoulos (GÉANT) (ed.), Tomasz Kuczyński (PSNC), Peter Lényi (MUNI), Nicolas Liampotis (GRNET) (ed.), Janos Mohacsi (Pro-M), Jan Pavlíček (MUNI), Wolfgang Pempe (DFN), Mischa Sallé (Nikhef), Mark van de Sanden (GÉANT), Gabriel Zachmann (KIT)

Version:	v1.0
Document Code:	AARC TREE D1.2
DOI:	N/A

Abstract

This document presents the final revision of the AARC Blueprint Architecture 2025 (AARC-BPA-2025), providing software architects and technical decision-makers with a modular framework for designing Authentication and Authorisation Infrastructures (AAIs) to support international research collaborations. Building on AARC-BPA-2019, it introduces a refined architecture based on Functional Capabilities - Identity Management, Collaboration Management, and Service Integration - while retaining the established five-layer component model. The capability-based perspective clarifies functional responsibilities, supports modular deployment patterns, and enables interoperable implementations across independent administrative domains. AARC-BPA-2025 incorporates updated trust models and reflects insights from real-world implementations. The revision is backwards compatible with AARC-BPA-2019 deployments and aligns with evolving policy and technical requirements within the international research ecosystem.

The AARC-TREE project is co-funded by the European Union under the HORIZON-INFRA-2023-DEV-01 call

Table of Contents

1. Introduction

The AARC Blueprint Architecture (BPA) provides a set of interoperable architectural building blocks for software architects and technical decision-makers who design and implement access management solutions for international research collaborations. The AARC BPA enables scalable, secure, and policy-aligned Authentication and Authorisation Infrastructures (AAIs), allowing users from diverse institutions and identity providers to access distributed digital resources across organisational, national, and disciplinary boundaries. The AARC BPA reflects practical lessons learned from large-scale deployments. Its modular design supports the integration of existing components and standards, encourages the reuse of best practices, and facilitates interoperability across heterogeneous infrastructures and service providers.

This document presents the final revision of the latest iteration of the architecture: **AARC-BPA-2025**. It builds on the previous version (AARC-BPA-2019 [AARC-G045]) by refining core concepts, updating architectural patterns, and aligning with evolving requirements from research infrastructures and the operational realities of international deployments.

AARC-BPA-2025 introduces a shift in how the architecture is described: rather than focusing solely on components, the architecture is now framed around clearly defined Functional Capabilities. These capabilities — such as Identity Management, Collaboration Management, and Service Integration — can be implemented in various combinations depending on the needs of a specific use case. This capability-driven view promotes architectural clarity, supports modular reuse, and accommodates both centralised and distributed deployment models.

The remainder of this document is organised as follows:

- Chapter 2 presents the AARC-BPA-2025, describing its Component Layers and Functional Capabilities.
- Chapter 3 describes how interoperable AARC-BPA-2025 implementations interact across administrative and organisational boundaries, including the Collaboration-first and User Identity-first approaches.
- Chapter 4 discusses federation models for interconnecting AARC BPA implementations.
- Chapter 5 outlines security and privacy considerations.
- Chapter 6 explores different approaches to allow AARC-compliant AAI services to establish trust relationships in the context of an identity federation to support OpenID Connect and OAuth 2.0-based interactions.
- Chapter 7 details the mechanism for cross-infrastructure validation of OAuth 2.0 access tokens using Proxied Token Introspection.
- Chapter 8 examines how emerging identity technologies, including Verifiable Credentials and Digital Identity Wallets, relate to the AARC-BPA.
- Chapter 9 concludes the document with a summary of key updates of this revision.

1.1. Terminology

Term	Definition
AAI Service	A service that enables authenticated and authorised access to resources. [AARC-G045]
Access Token	A credential represented by a string, issued to a client by an authorization server, and used to access protected resources. (see also [RFC6749])
Attribute	An attribute is a named property associated with an entity. An attribute is controlled vocabulary that is semantically in common to both the Community and the service and resource providers who are supposed to interpret these attributes correctly.
Attribute Authority (AA)	An attribute authority is a technical unit controlled by an organisation (such as a community or infrastructure) that is entitled to make certain statements about entities, and assign attributes to them. The AA may be the authoritative source of these statements, or be a proxy that asserts these statements based on trusted information it obtains from other sources. The assertions/attributes carrying these statements are generated by the technical service of the AA Operator.
Authorization Server (AS)	The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization. (see [RFC6749])
Client	An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices). (see [RFC6749])
Collaboration	A dynamic collection of individuals, institutions and resources which engage in flexible, secure, and coordinated resource sharing. In the context of the AARC BPA, “Community” and

	"Collaboration" are architecturally the same and the terms are therefore used interchangeably In this document.
Collaboration Management Functional Capability	The AARC-BPA-2025 Functional Capability that enables the management of collaborations, including user membership in groups/projects, assignment of roles, and user access to services within the scope of their collaboration.
Collaboration Management Service	An AAI service that implements the Collaboration Management Functional Capability.
Community	A group of Users, organised with a common purpose, and jointly granted access to the Collaboration. It may act as the interface between individual Users and the Collaboration (see [AARC-I082]). In the context of the AARC BPA, "Community" and "Collaboration" are architecturally the same and the terms are therefore used interchangeably In this document.
Community AAI	An AAI service that also enables the use and management of community identities for access to resources. It comprises three (3) AARC BPA-2019 Component Layers: the Access Protocol Translation, the Community Attributes Services, and the Authorisation. [AARC-G045]
Entity	Federation Entity. Something that has a separate and distinct existence and that can be identified in a context. (see [OID-Fed])
Entity Identifier	A globally unique URL that is bound to one Federation Entity. This URL MUST use the https scheme, have a host component, and MAY contain port and path components. It MUST NOT contain query parameter or fragment components. (see [OID-Fed])
Federation	Identity Federation. An association of organisations that come together to securely exchange information as appropriate about their users and resources to enable collaborations and transactions.

Identity Management Functional Capability	The AARC-BPA-2025 Functional Capability that encompasses the authentication and identity lifecycle management of users. It integrates with external Identity Providers and supports identity enrollment and linking across multiple authentication sources. It establishes persistent user identifiers and supports signalling and/or step-up of authentication strength (quality of authentication) and identity assurance (quality of identity) to downstream services.
Identity Management Service	An AAI service that implements the Identity Management Functional Capability.
Infrastructure	The IT hardware, software, networks, data, facilities, processes and any other elements that together are required to develop, test, deliver, monitor, control or support services
Infrastructure Proxy	An AAI service of a research infrastructure or e-Infrastructure (hereafter termed infrastructure) that enables access to resources offered by Service Providers connected to that infrastructure. This AAI service comprises two (2) AARC BPA-2019 component layers: the Access Protocol Translation and the Authorisation. [AARC-G045]
OpenID Provider (OP)	OAuth 2.0 Authorization Server that is capable of Authenticating the End-User and providing Claims to a Relying Party about the Authentication event and the End-User. (see [OIDC-Core])
Pairwise Identifier	An identifier associated with a subject intended for a specific relying party (or group of parties) and not for anyone else. (see also [SHIB-NameIds])
Persistent Identifier	An identifier associated with a subject intended to be used for an extended period of time across multiple sessions. This is the opposite of a Transient Identifier. (see also [SHIB-NameIds])

Public Identifier	An identifier associated with a subject that is not specific to any particular relying party or group of relying parties, allowing it to be shared across multiple relying parties. (see also [SHIB-NameIds])
Relying Party (RP)	A consumer of the attributes, who has to establish a trust relationship with the entity providing it with the attributes.
Resource Server (RS)	The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens. (see [RFC6749])
Service	A service implements an RP to consume attributes, based on which authorisation decisions are made. Services are named differently for distinguishing which AAI component they are connected to. (The naming also indicates the available user base.)
Shared Identifier	See Public Identifier
Service Integration Functional Capability	The AARC-BPA-2025 Federation Capability that ensures interoperability between the Authentication and Services Component Layers by performing protocol mediation, identity translation and enrichment, and access policy enforcement. It may be implemented in Infrastructure Proxies, Site Local Infrastructure Proxies, or directly within Services.
Service Integration Service	An AAI service that implements the Service Integration Functional Capability.
Subject	A person, organisation, device, hardware, network, software, or service.
Transient Identifier	An identifier associated with a subject intended to be used for a relatively short period of time that need not span multiple sessions (see also [SHIB-NameIds]).

Trust Anchor	An Entity that represents a trusted third party. (see [OID-Fed])
Trust Authority	A third party which is trusted to identify entities. • In OpenID Federation, a Trust Authority is a third party represented by one or more Trust Anchors. • In IGTF, a Trust Authority is a Certification Authority; the Trust Anchors are its root or intermediate certificates in the IGTF distribution. • In other models, a Trust Authority could simply maintain and publish a list of identifiers of entities, provided the association between each entity and its identifier can be checked by other trusted means.
Trust Mark	Statement of conformance to a well-scoped set of trust and/or interoperability requirements as determined by an accreditation authority. Each Trust Mark has a Trust Mark identifier. (see [OID-Fed])
Trust Mark Issuer	A Federation Entity that issues Trust Marks. (see [OID-Fed])

1.2. Notational Conventions

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in [RFC2119].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

2. Architecture Component Layers & Functional Capabilities

This chapter presents the Component Layers and Functional Capabilities of the 2025 revision of the AARC Blueprint Architecture (AARC-BPA-2025). A comparison with the previous version, AARC-BPA-2019, is included in Annex A to highlight key updates and refinements.

2.1. Component Layers

AARC-BPA-2025 includes five (5) Component Layers: ***Authentication, Attribute Services***, ***Access Protocol Translation***, ***Authorisation*** and ***Services.***

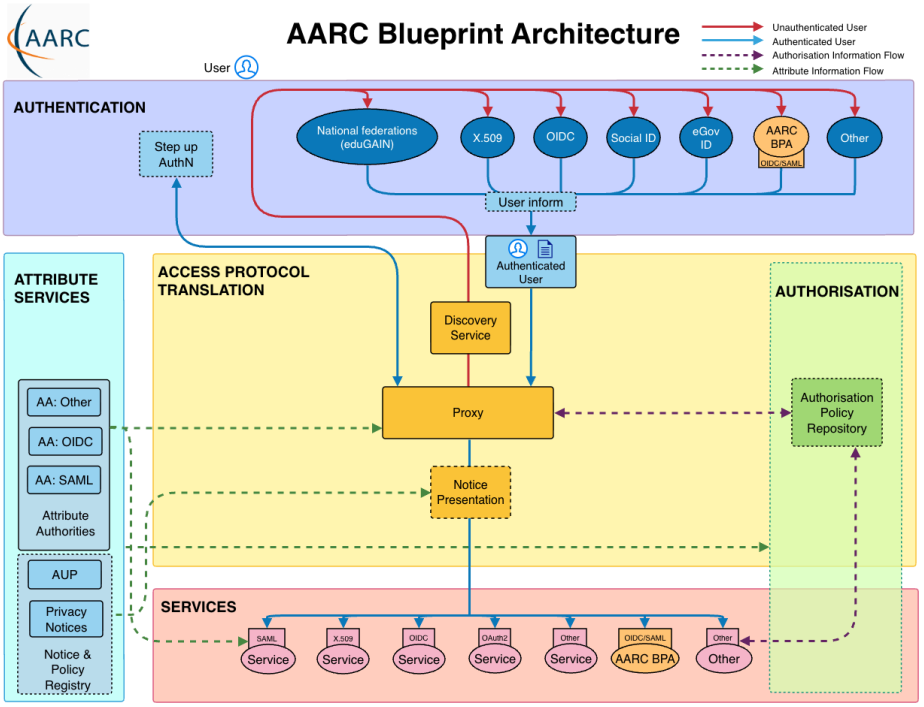


Figure 2.1: Component Layers of the AARC Blueprint Architecture 2025 (AARC-BPA-2025)

As illustrated in Fig. 2.1, each of these layers includes one or more logical components, grouped by their functionality:

- **Authentication Layer** - Contains services for the identification and authentication of users[1]. For existing implementations in the research and education space, these services typically include **Identity Providers** using the Security Assertion Markup Language (SAML),

[1] The term "users" does not exclude other types of subjects, such as agents, robots, etc.

certification authorities, and, more recently, identity providers using OAuth 2.0-based login flows, typically based on OpenID Connect (OIDC). Additional **AARC-compliant Proxies** may also be present to enable chaining of AAI services using OIDC or SAML[2]. The Authentication Layer may also include supporting components such as **Step-Up Authentication** mechanisms to increase authentication strength when required, and **User Inform** mechanisms for presenting notices or obtaining consent related to the release of personal data.

- **Attribute Services Layer** - Groups components related to managing and providing information (attributes) about users. For instance, this information can include group memberships and roles in the context of a collaboration as well as collaboration-specific user identifiers, which are added to the information that is already provided directly by the identity providers from the Authentication Layer. The Attribute Services Layer also includes a **Notice & Policy Registry**, which maintains unique identifiers (URIs) and associated metadata for policies such as Acceptable Use Policies (AUPs), Privacy Notices, and Terms of Service, in order to track user acceptance and support the notice presentation models (e.g. machine-readable aggregated notices, common notice, or cascading policy) defined in [AARC-G083].

- **Access Protocol Translation Layer** - Addresses the requirement for supporting multiple authentication and authorisation technologies. It includes the following services:
    - **SP-IdP-Proxy (Proxy),** which serves as a single integration point between the Identity Providers from the Authentication Layer and the Service Providers in the Services Layer. Thus, the Proxy acts as a Service Provider (SP) towards the Identity Providers, while towards the Services it acts as an Identity Provider (IdP). The proxy is also responsible for credential and protocol translation - where necessary.
    - **Discovery Service,** which enables the selection of the user's authenticating IdP.
    - **Notice Presentation Component (NPC),** which is responsible for presenting relevant notices to end-users. It ensures that end-users are informed regarding the processing of their personal data and are able to acknowledge policies such as Acceptable Use Policies (AUPs). The NPC also allows for recording and tracking user acceptance of such notices through the Notice & Policy Registry. When required, the NPC can signal accepted policy identifiers to downstream services, in accordance with the notice presentation models defined in [AARC-G083].

- **Authorisation** - Contains components for managing and implementing policies to authorise access to services. An **Authorisation Policy Repository** may be utilised. Since authorisation can occur at the Proxy level, at the service level, or at both, the Authorisation Layer overlaps with the Services Layer.

- **Services Layer** - Contains the services users want to – directly or indirectly – use. Access to these services is protected (possibly using different technologies). These services can range from simple web-browser-based services, such as wikis or portals for accessing computing and storage resources, to non-web-browser-based resources such as APIs, login shells, or workload management systems. Access may be interactive and non-interactive and could be

[2] Using SAML-only to connect AARC-compliant Proxies is generally suitable for use cases limited to web browser–based Single Sign-On (SSO); delegated access or offline access would require an OIDC / OAuth 2.0-based connection.

delegated via / to additional services, such as **AARC-compliant Proxies**, allowing users to access resources provided by other infrastructures through their respective Proxies.

2.2. Functional Capabilities

The AARC-BPA-2025 introduces the concept of Functional Capabilities to complement the established component layers defined in Section 2.1. Each Functional Capability represents a cohesive set of functions that collectively support a specific aspect of federated identity and access management. These Functional Capabilities span one or more component layers and define the logical responsibilities and service groupings required to deliver secure, scalable, and interoperable access to research infrastructures.

This Functional Capability-based view supports modular implementation, clearer governance, and improved alignment with community and infrastructure needs. Fig. 2.2 illustrates the three key Functional Capabilities — ***Identity Management***, ***Collaboration Management***, and ***Service Integration*** — highlighting their respective functions.

Please note that in the Functional Capability view, the Community AAI defined in AARC-BPA-2019 encompasses two Functional Capabilities, namely the Identity- and Collaboration Management capabilities (see Annex A: Revisions since AARC-BPA-2019). They still may be provided through a single technical implementation, or as separate ones. Similarly, the Infrastructure Proxy has now been generalised as the Service Integration Functional Capability but remains backwards compatible.

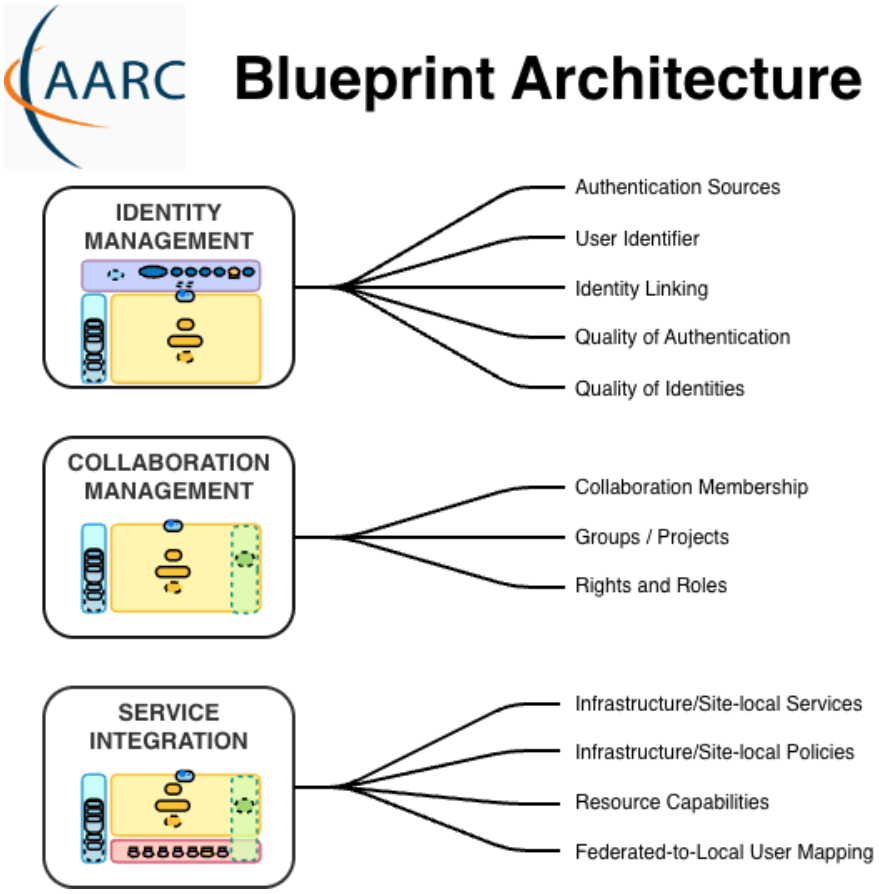


Figure 2.2: The three key Functional Capabilities of the AARC Blueprint Architecture 2025 (AARC-BPA-2025) highlighting their respective functions

- **Identity Management Functional Capability -** Encompasses the authentication and identity lifecycle management of users. It integrates with external Identity Providers and supports identity enrollment and linking across multiple authentication sources. It establishes persistent user identifiers and supports signalling and/or step-up of authentication strength (quality of authentication) and identity assurance (quality of identity) to downstream services. This capability includes components from the Authentication layer, the Access Protocol Translation layer, and the Attribute Services layer.

- **Collaboration Management Functional Capability** - Enables the management of collaborations, including user membership in groups/projects, assignment of roles, and user access to services within the scope of their collaboration. It builds upon identity information from the Identity Management and facilitates collaboration-driven authorisation logic. This capability includes components from the Attribute Services layer, the Access Protocol Translation layer, and the Authorisation layer.

- **Service Integration Functional Capability** - Provides protocol mediation and ensures interoperability between the Authentication and the Services layers. It translates identity assertions, enriches identities and manages access requests. This capability includes components from the Attribute Services layer, the Access Protocol Translation layer and the Authorisation layer.
    - The Service Integration Functional Capability may be implemented in **Infrastructure Proxies** which connect a large number of services, translate identity assertions, and enrich identities with infrastructure-specific attributes (e.g. resource capabilities, infrastructure roles).
    - The Service Integration Functional Capability may also be implemented in **Site Local Infrastructure Proxies** which focus on connecting multiple services at a site, on ensuring that federated identities and access policies can be effectively integrated at site level, and on enforcing site-specific policies, such as attribute transformation or access filtering, enabling integration of federated users into local environments.
    - Finally, The Service Integration Functional Capability may also be implemented directly in **Services** that natively support the protocols and policies of the Layers they connect to.

3. Architecture for interoperable AARC BPA implementations

This chapter describes how AARC-BPA-2025 compliant implementations interoperate across administrative and organisational boundaries. It explains how the Functional Capabilities introduced in Section 2.2 — Identity Management, Collaboration Management, and Service Integration — interact to support secure and scalable access to services.

Interoperability in the AARC BPA is achieved through a clear separation of concerns:

- **Identity Management** establishes and maintains a persistent user identifier, independent of collaboration context.
- **Collaboration Management** manages collaboration-specific membership, roles, and authorisation attributes.
- **Service Integration** provides a single, controlled access gateway for services within an infrastructure.

When a user accesses a service, the Service Integration acts as the integration point and delegates authentication to a trusted Identity Management service. Identity establishment is therefore decoupled from collaboration-specific attribute management. Where required, collaboration context is resolved through interaction with the appropriate Collaboration Management service. The resulting identity and authorisation attributes are aggregated before being returned to the requesting service.

This Functional Capability-based interaction model enables:

- Consistent identity representation across infrastructures
- Reuse of identity management services by multiple collaborations
- Controlled attribute aggregation and release

The AARC-BPA-2025 revision does not deprecate the architectural model defined in AARC-BPA-2019.

The capability-based description introduced in AARC-BPA-2025 is an architectural clarification and refinement of the existing model. Implementations conforming to AARC-BPA-2019 remain valid and interoperable. In particular:

The five component layers remain in scope and function.

The former Community AAI model is preserved in functional terms and can still be implemented as a single service implementing the Identity Management and Collaboration Management functional capabilities.

No existing deployment pattern defined in AARC-BPA-2019 is invalidated.

The introduction of Functional Capabilities (Identity Management, Collaboration Management, and Service Integration) provides a clearer separation of concerns and supports modular deployment models, but it does not impose new mandatory architectural constraints on existing compliant implementations. Refer to Annex A for a more detailed description of the revisions since AARC-BPA-2019.

The following sections detail the interaction patterns between these Functional Capabilities, including identity establishment, optional collaboration context resolution, attribute aggregation, and service access in both single- and multi-infrastructure environments.

3.1. Collaboration-first approach

In the Collaboration-first approach (see Fig. 3.1), the collaboration boundary acts as the primary integration point for access to services.

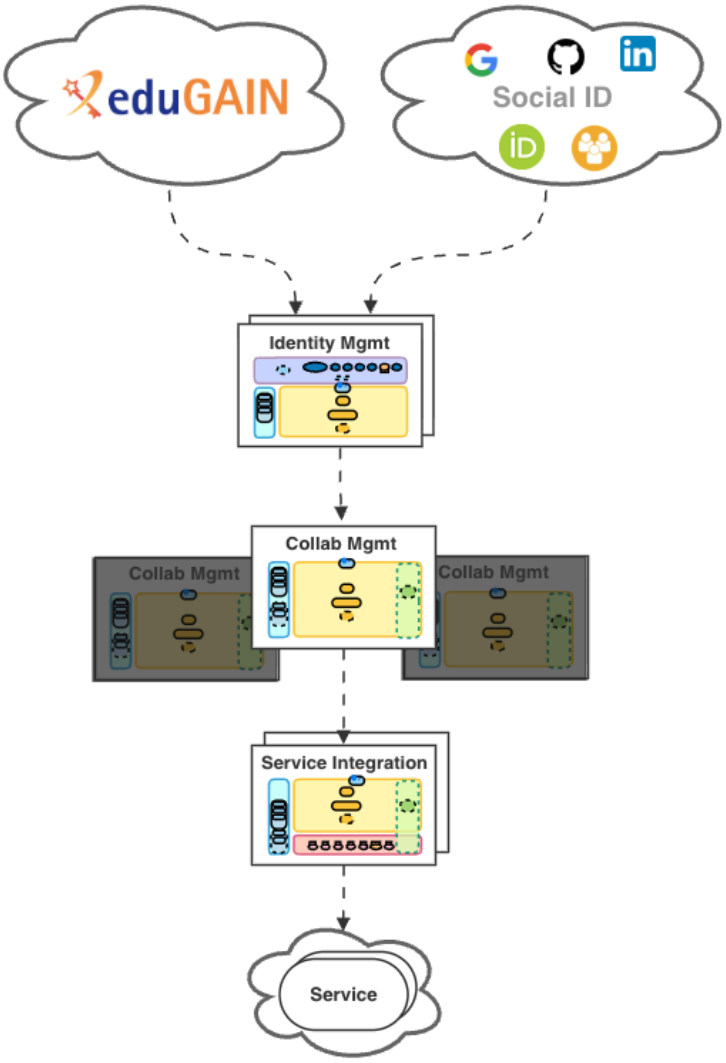


Figure 3.1: AARC-BPA-2025 Collaboration-first approach

Access to services is initiated through one or more AAI services implementing both the Identity Management and Collaboration Management capabilities. These services:

- Authenticate the user through Identity Providers of the Authentication layer;
- Establish a persistent subject identifier;
- Associate the user with collaboration-specific context;
- Provide collaboration-specific attributes (e.g. group membership and role information, or resource capabilities) to connected services.

In this model, collaboration context is established at authentication time. Services connected to the collaboration consume identity and authorisation information scoped to that collaboration.

The Identity Management and Collaboration Management capabilities MAY be provided by a single AAI service or by distinct services. In the latter case, the Identity Management Functional Capability MAY be provided externally. In either case, the functional responsibilities and externally visible interfaces remain unchanged.

The Service Integration capability connects collaboration-scoped user identities to services within an infrastructure domain. A Service Integration service MAY connect to multiple collaborations and accept identities issued by distinct Identity and Collaboration Management services. Services within the infrastructure domain rely on the Service Integration service as a single trusted integration point, independent of the number of upstream collaborations.

3.1.1. Relationship to AARC-BPA-2019

The Collaboration-first approach corresponds to the Community-first approach defined AARC-BPA-2019, where:

- The functions implemented by the former Community AAI are now represented by the Identity Management and Collaboration Management Functional Capabilities.
- The former Infrastructure Proxy corresponds to the Service Integration Functional Capability.

AARC-BPA-2025 does not alter this deployment pattern. The introduction of Functional Capabilities clarifies the separation of responsibilities without redefining the underlying architecture.

3.2. User Identity-first approach with dynamic collaboration context

Similar to the Collaboration-first approach, in the User Identity-first approach (see Fig. 3.2), the process is initiated when a user accesses a Service. It delegates authentication via an AAI Service implementing the Service Integration Functional Capability.

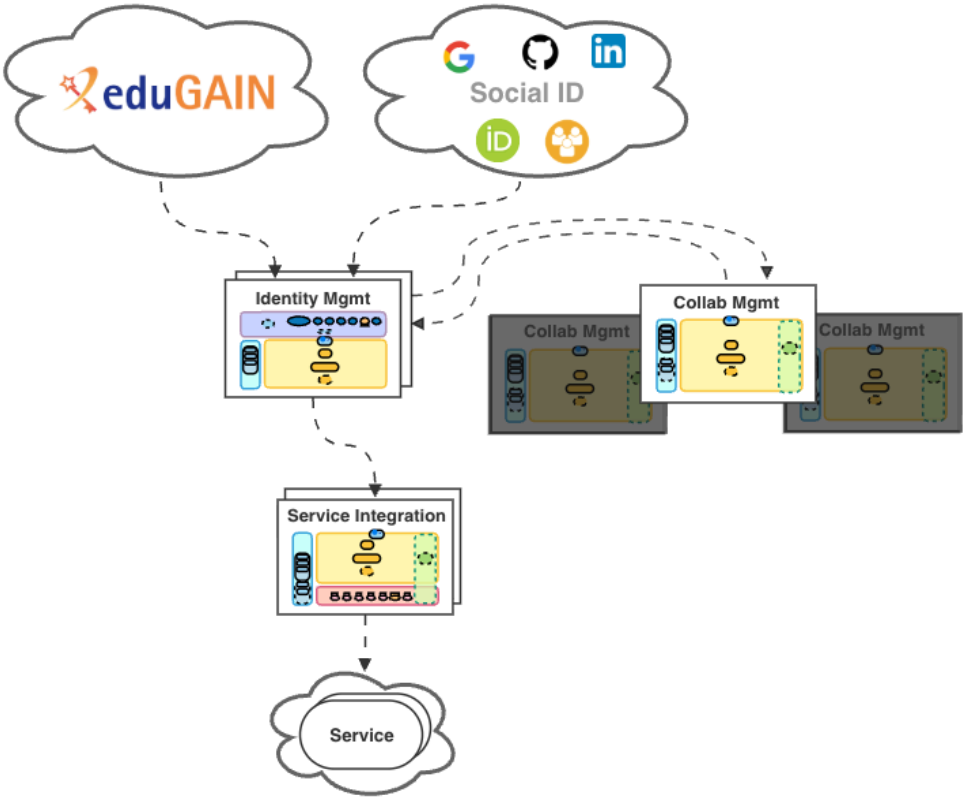


Figure 3.2: AARC-BPA-2025 User Identity-first approach with dynamic collaboration context

The Service Integration service redirects the user to a trusted Identity Management service. The Identity Management service:

- Authenticates the user through Identity Providers of the Authentication layer;
- Establishes a persistent subject identifier;
- Maintains identity information independent of any specific collaboration.

In this model, collaboration context is not directly established at authentication time. If the requesting Service requires authorisation based on collaboration membership, the Identity Management service retrieves the relevant collaboration-specific attributes from the appropriate Collaboration Management service. The applicable collaboration context MAY be:

- Selected by the user,
- Indicated by the requesting Service, or
- Determined according to policy enforced by the Identity Management service.

Where no collaboration-related information is required, the Collaboration Management Functional Capability is not invoked.

The Identity Management service aggregates the persistent subject identifier with any retrieved collaboration-specific attributes and returns the resulting identity representation to the Service Integration service. The Service Integration service forwards this information to the requesting Service.

Services rely on the Service Integration service as their trusted integration point, independent of whether collaboration context was involved in the authentication process.

The Identity Management and Collaboration Management capabilities MAY be implemented by a single service or by distinct services. Dynamic retrieval of collaboration attributes does not alter the externally visible interfaces of the architecture.

3.2.1. Relationship to the Collaboration-first approach

The User Identity-first approach differs from the Collaboration-first approach in the point at which collaboration context is established. In the User Identity-first model, identity authentication precedes collaboration resolution. In the Collaboration-first model, collaboration context is inherent in the authentication session.

Both interaction models rely on the same Functional Capabilities and are interoperable within AARC-BPA-2025.

4. Models for federating AARC BPA Implementations

Different federation models may be employed depending on the number and distribution of AAI services to be interconnected.

For a limited number of AAI services, bilateral trust relationships may be sufficient. However, bilateral configurations do not scale well as the number of participating entities increases, due to the operational overhead of maintaining pairwise trust agreements.

For larger federations, mesh or hub-and-spoke models — where a central service provides shared functions such as identity establishment, trust coordination, or service integration — may improve scalability and simplify operational trust management.

OpenID Federation [OID-Fed] introduces a distributed trust chain model that enables scalable and decentralised trust establishment. This model aligns with the trust profiles defined in [AARC-G100] and supports dynamic trust evaluation. However, the operational deployment of OpenID Federation is still evolving.

The choice of federation model affects trust establishment and operational scalability but does not alter the Functional Capabilities defined in AARC-BPA-2025.

5. AARC-BPA-2025 Security and Privacy Considerations

AARC BPA implementations should allow the selective release of attributes, following the principle of data minimisation.

In the User Identity-first approach, the user identifier is issued by the Identity Management service. If there are privacy or regulatory requirements that need to be met, the Identity Management service should not release the shared user identifier to certain relying parties, where this could enable cross-service correlation of user activity (e.g. tracking or profiling). In such cases, pairwise identifiers should be considered.

6. Trust Establishment using OpenID Federation

This chapter presents the model for establishing trust between AARC-compliant AAI services using OpenID Federation [OID-Fed]. This model is based on the guideline document AARC-G100, which defines the required behaviour of Proxies, Trust Authorities, and Trust Mark Issuers, as well as two trust profiles (Basic and Fine-Grained).

AARC-G100 has been shared with and discussed within the AEGIS community and has been piloted through the Validation Work Package (WP4). Validation results, documented in AARC TREE Deliverable D4.1, identify the need for improved software support and federation tooling.

6.1. Trust Model

In a federated AAI system, each trust domain — typically representing an administrative, legal or technical boundary — hosts one or more AARC-compliant AAI services.

AARC-compliant AAI services incorporate an SP-IdP-Proxy component [AARC-G045], acting as both a Service Provider and an Identity Provider. In the context of OIDC, these roles correspond to a Relying Party and an OpenID Provider: the proxy acts as a Relying Party towards its Identity Providers or upstream AAIs, and as an OpenID Provider towards its Service Providers. To participate in Research and Education (R&E) identity federations, these proxies often require a protocol translation layer (e.g. OIDC to SAML2). The federation model described in this document enables direct participation of OIDC and OAuth 2.0-based entities in federations, eliminating the need for such a translation, by establishing trust through a trusted third party known as a Trust Authority. A Trust Authority is an entity whose primary role is to issue statements about other entities — such as OAuth 2.0 Authorization Servers (AS) and Resource Servers (RS) — that participate in a federation.

The trust model is based on mechanisms from the OpenID Federation Specification [OID-Fed] and enables entities that have no direct trust relationship to establish trust by constructing a Trust Chain (trust path) from the other entity to a trusted third party — a Trust Anchor.

Figure 6.1 shows a single Trust Authority for simplicity, but the trust evaluation mechanisms discussed are designed to support more complex federation topologies, e.g. with multiple Trust Authorities, hierarchical federation structures, or even multiple overlapping federations.

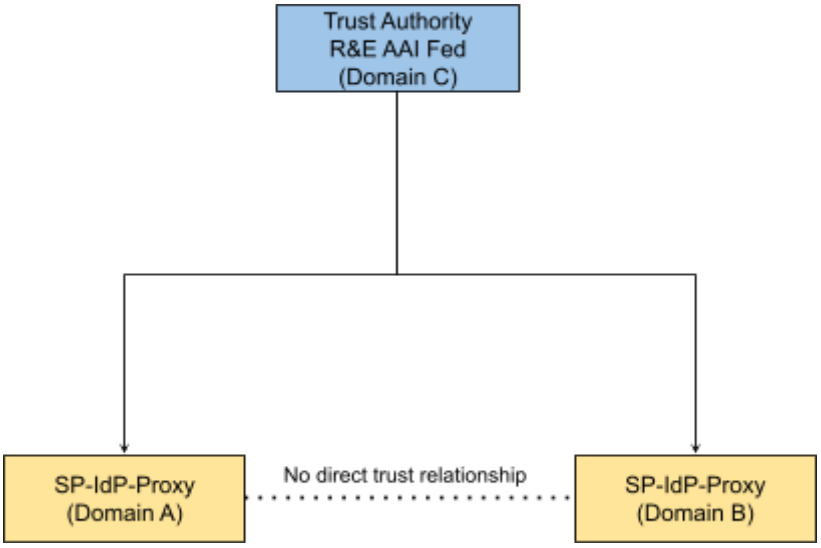


Figure 6.1: Simple federation model based on the use of a common Trust Authority.

While a **Trust Authority** is defined as a third party which is trusted to identify entities — by issuing statements about them — it can take on different roles in a federated environment. For example, from the point of view of a Proxy:

- an **Immediate Superior Entity** is a Trust Authority that represents a federation operator which:
    - onboards the Proxy into its federation through an out-of-band registration process; and,
    - is able to issue statements about the Proxy.

  A Proxy can have multiple Immediate Superiors, i.e. can be part of multiple federations

- a **Trust Anchor** is a Trust Authority that is inherently trusted by the proxy on the basis of its keys.
    - It does not need to onboard the proxy or trust the proxy, be it directly or indirectly.
    - A Proxy can have multiple Trust Anchors.

A more complex example assumes that proxies can be part of multiple federations (have multiple Immediate Superiors) and trust multiple Trust Anchors. In addition, trust hierarchies are possible, where Trust Authorities can register with other Trust Authorities to be part of their federations.

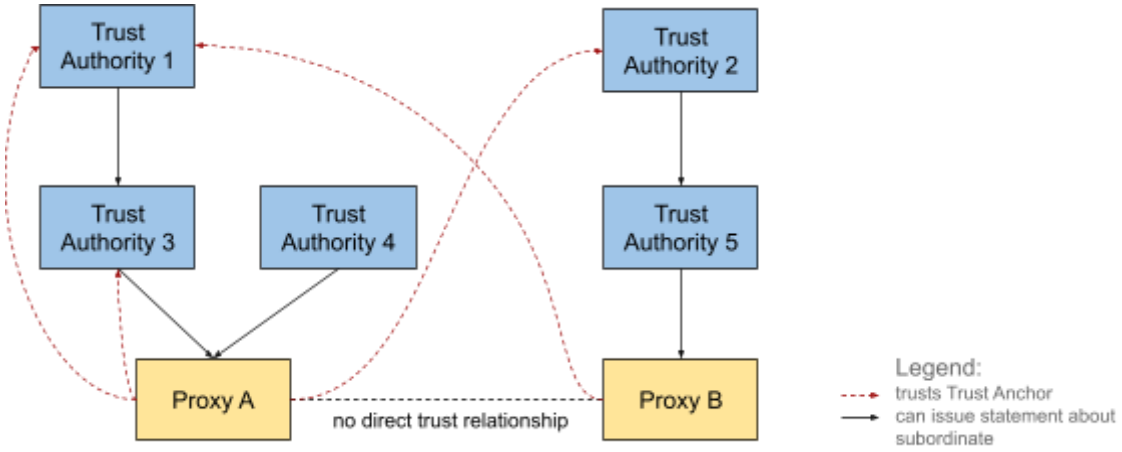


Figure 6.2: A more complex trust model with multiple Trust Authorities, hierarchies, and asymmetric trust.

In the example in Fig. 6.2, the black arrows depict which entities a Trust Authority can issue statements about. As such, we have:

- Proxy A's Immediate Superiors: Trust Authority 3, Trust Authority 4
- Proxy B's Immediate Superiors: Trust Authority 5

In addition, each proxy has its own Trust Anchors, for example:

- Proxy A's Trust Anchors: Trust Authority 3, Trust Authority 1, Trust Authority 2
- Proxy B's Trust Anchors: Trust Authority 1

In this example, Proxy A and Proxy B:

- don't have a direct trust relationship
- can establish trust by constructing a trust path from the other entity to one of their Trust Anchors:
    - Proxy A trusts Proxy B because there is a path from Proxy B to Trust Authority 2
    - Proxy B trusts Proxy A because there is a path from Proxy A to Trust Authority 1

In the real world, a more fine-grained approach to trust might be needed.

- Registering into a federation may be conditioned by compliance with certain federation policies.
- This set of policies may need to be signalled to other proxies that are not part of the same federation in a consistent way, so that proxies can additionally decide to trust other proxies on the basis of these policies.

6.2. OpenID Federation

This section describes the building blocks of OpenID Federation that are relevant for the trust establishment in the context of this document. This section is not normative.

OpenID Federation provides a scalable framework for establishing trust relationships without pre-existing bilateral agreements. Unlike traditional federations based on manual metadata exchange (e.g. SAML based), OID-Fed allows federation participants to:

- dynamically discover each other's metadata
- dynamically evaluate trust by constructing a verifiable trust chain to a trusted third party
- assert conformance to federation policies using cryptographically verifiable Trust Marks

Federations in OID-Fed are modelled as a directed graph of trust relationships, typically described as a **"tree"**, though it may more often resemble a **"mesh"** in more inter-connected, multi-anchor federations.

- **Leaf entities** represent federation participants (e.g. SP-IdP-Proxies).
- **Intermediate entities** and **Trust Anchors** represent federation operators, which issue signed statements about their members.
- **Trust Anchors** are trusted third parties whose (public) signing keys are distributed out-of-band and treated as the basis of trust.

The basis for the OID-Fed trust framework is the **Entity Statement**, a signed JWT using asymmetric cryptography.

- Entities publish their metadata as self-signed entity statements called **Entity Configurations**, at a well-known URL.
- Trust Anchors and Intermediate entities issue signed entity statements called **Subordinate Statements** about their immediate subordinates (containing the subordinate's public key) as a way to assert that the subordinates are members of their federation.
- A set of entity statements can form a path from a leaf entity to a Trust Anchor, called a **Trust Chain**. The Trust Chain can be cryptographically verified using the signature of each statement, with the Trust Anchors's key inherently trusted.
- An entity A can trust an entity B if it can construct a valid trust chain from B to one of A's Trust Anchors.
- Additionally, subordinate statements can contain so-called **Metadata Policies**, which allow Intermediate entities to enforce technical constraints on their subordinates' metadata. This enables a federated resolution of metadata by aggregating metadata policies in a trust chain and applying them to a leaf entity's metadata. Note that metadata policies, as defined by OID-Fed, are different from federation policies, which are more general guidelines and rules that govern a federation, and the interaction and operation of entities within a federation.

Even though not in the scope of the OID-Fed spec, the process of registering entities with their superiors (onboarding in the federation) is a prerequisite for creating the trust fabric. Each federation operator is responsible for providing such a process to exchange keys and verify the eligibility of an entity to join their federation, before being able to issue an entity statement about it. In turn, federation members signal membership by publishing their immediate superiors in their own entity configuration (via authority_hints).

6.2.1. Client Registration

The OID-Fed specification defines two methods that use Trust Chains to establish trust between an RP and an OP that have no prior explicit configuration or registration between them: Automatic Registration and Explicit Registration. Both methods can also be used for OAuth 2.0 profiles other than OpenID Connect.

For a successful communication, the RP and OP MUST support a common client registration method. Therefore, federations SHOULD agree on the supported client registration methods. The table below shows a comparison of the two methods.

	Automatic registration	**Explicit registration**
Mechanism	• RP can make authentication requests to the OP with no prior registration • RP uses its federation entity id as the client ID in all interactions • OP must dynamically discover and validate trust with RP on the fly, then client is automatically accepted • Must use asymmetric cryptography to authenticate requests since there is no client secret assigned	• Dedicated client registration similar to OIDC, but RP submits an Entity Configuration or a whole Trust Chain instead of just metadata • OP validates trust chains and applies registration policies • Once registration is completed, client id and secret are assigned and subsequently regular OpenID authentication requests can be done
Pros	• No need for prior contact • Fully dynamic	• More control • Allows policy checks and pre-registration processing • Allows the use of existing OIDC libraries without OID-fed support for the OIDC communication.
Cons	• Less initial control, policies evaluated at runtime • Must be able to validate on the fly	• Requires client-side registration logic (limited scalability). • Client registration expires; client must be able to handle (periodical) re-registration.
Use cases	• Large federations with many potential clients	• Federations with specific metadata requirements

	• Dynamic trust for short-lived integrations	• Clients where the OIDC communication cannot / should not be changed.

6.2.2. Trust Marks

OID-Fed also introduces the concept of **Trust Marks**, which can be used to represent conformance to federation policies in a scalable and verifiable way. A Trust Mark, represented as a JWT, is a signed statement of conformance to a well-scoped set of criteria as determined by an accreditation authority, issued and signed by that accreditation authority. Trust Marks are published by an entity in their Entity Configuration, and they can be validated with a trusted accreditation authority, also called Trust Mark Issuer (recognised Trust Marks and their authorised accreditation authorities within the federation should be published by the Trust Anchor). The concept of Trust Mark delegation is also supported (via signed delegation JWTs), for cases where a Trust Mark Owner delegates the issuance of Trust Marks to (one or multiple) Trust Mark Issuers, for various administrative or technical reasons.

Trust Marks might be used in several scenarios:

- Local policies at the proxy: during trust evaluation between leaf entities, in addition to having a valid trust chain, entities might impose additional constraints to establish trust, e.g. entities must conform to specific policies as represented by Trust Marks.
- Federation onboarding: Trust Authorities might require entities to have specific Trust Marks in order to be allowed to join a federation; also, Trust Authorities can issue a membership Trust Mark after registration.
- Research Community onboarding: A certain research community may use a Trust Mark Issuer to issue Trust Marks to certain Entities who are part of a specific collaboration. Service providers who are members of this collaboration can use the Trustmark as part of the access rights evaluation, as it is issued by the research Community itself, rather than delegated via e.g. a National Federation
- Entity discovery: Trust Marks can be used to filter entities during discovery processes.

It is important to distinguish between Trust Chains and Trust Marks: they are orthogonal concepts that serve different but complementary purposes. Trust Anchors are used to validate identity, federation membership and metadata integrity, while Trust Marks are used for policy enforcement and authorization filtering. Essentially, a Trust Anchor specifies who the entity is and where it fits in the federation (an “inventory”), while the Trust Mark asserts what the entity complies with (a “filter”).

6.3. Establishing trust using OID-Fed in the context of the AARC BPA

This section maps the building blocks of OID-Fed to address the trust requirements between Proxies.

Consider the Authorisation Server (AS) and the Client roles. As depicted in Fig. 6.3, for an SP/IdP Proxy A to trust a Proxy B, the client interface of Proxy A (Client_A) needs to trust the AS interface of Proxy B (AS_B). To replace the current manual approach with OpenID Federation, several different options exist.

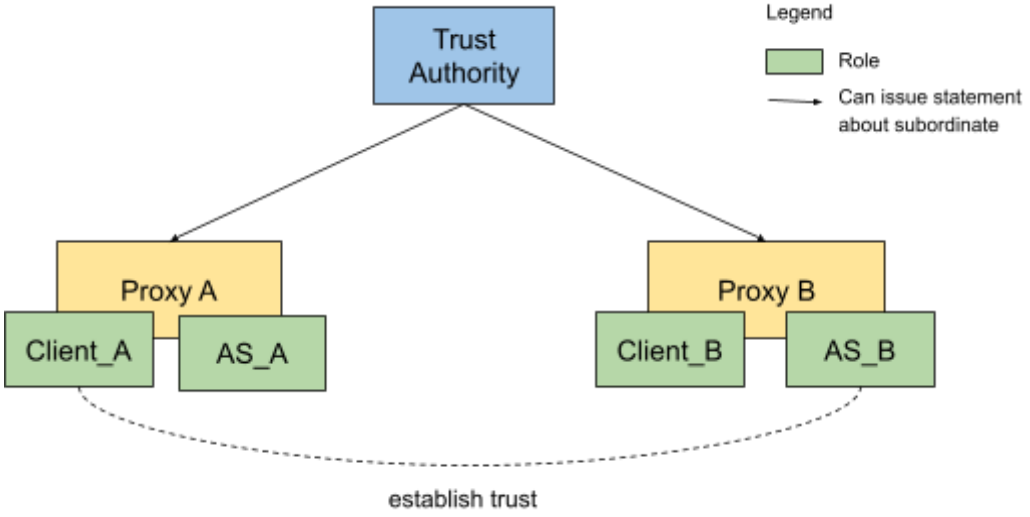


Figure 6.3: Establishing trust between two proxies in a simple trust model with one Trust Authority. The proxies can have multiple roles (Client, Authorisation Server). For Proxy A to trust Proxy B, Client_A needs to trust AS_B.

AARC-G100 defines two different profiles for the trust model that can enable trust between the two proxies:

1. **[G100.1] Basic trust model:** this profile defines the minimum set of requirements needed to establish trust, based only on registration of proxies with superior entities.
2. **[G100.2] Fine-grained trust model:** this profile adds an additional layer to the basic trust model for finer grained trust, by using Trust Marks and metadata policies to explicitly express federation policy requirements.

Compliance with AARC-G100 can be expressed as a two-step process of compliance with these two profiles.

6.3.1. G100.1 Basic Trust Model

This section is normative.

Prerequisites:

- Proxies MUST join the trust fabric by registering with Trust Authorities.
- Both proxies MUST be able to establish a trust chain for the other party.

Notes:

- In this model, requirements for federation policies are implicit, since they are defined and considered out-of-band in the onboarding with Trust Authorities. When a Proxy trusts a Trust Anchor, it implicitly acknowledges all the policies required at the Trust Anchor.

To establish trust with another Entity (AS_B) the following MUST be done (high level):

1. Client_A uses OID-Fed logic to verify AS_B has a valid Trust Chain (Sec 10 of OID-Federation Spec) to a Trust Anchor trusted by Client_A.

6.3.2. G100.2 Fine-Grained Trust Model

This section is normative.

Prerequisites:

- Basic Trust Model (G100.1).
- To indicate compliance with policies and policy profiles, both parties MUST use Trust Marks.
    - one Trust Mark per federation policy may make sense (e.g. [RAF], REFEDS Data Protection Code of Conduct [DPCoCo], [SIRTFI])
    - one Trust Mark per profile may make just as much sense (e.g. EOSC-AAI). Such a profile would be similar to a profile in RAF, e.g. RAF Cappuccino.
    - Both concepts can be used at the same time.
    - This requires Trust Mark Owners to issue Trust Marks, or to delegate this to Trust Mark Issuers that issue Trust Marks to the correct entities.
- Proxies MUST be able to apply Metadata Policies coming from upstream Trust Authorities during the Trust Chain Resolution Process.

To establish trust with another Entity (AS_B) the following MUST be done (high-level):

1. Client_A uses OID-Fed logic to verify AS_B has a valid Trust Chain (Sec 10 of OID-Federation Spec) to a Trust Anchor trusted by Client_A.
2. Client_A uses OID-Fed logic to apply Metadata Policies during the Trust Chain Resolution process.
3. Client_A verifies that AS_B has a valid Trust Mark for all the Trust Mark Types Client_A requires.

6.4. Trust Establishment (technical flow)

This section and all its subsections are normative.

In the following sections, features and processes that pertain only to the fine-grained trust model are denoted with [G100.2] and compliance with these is not required in the Basic model [G100.1]. Optional features in any of the trust models will additionally be denoted with “Optional”.

6.4.1. Onboarding process

To be able to establish trust, Proxies must already be part of the trust fabric. Proxies become part of the trust fabric by registering with a Trust Authority. The registration process is Trust Authority specific, and as such this guideline cannot mandate a specific process.

A typical process is depicted in Fig. 6.4 and consists of the following steps:

- Initiate the onboarding process in some way
- Perform public key exchange between the Proxy and the Trust Authority
- Trust Authority performs eligibility checks, which SHOULD include:
    - Proxy has published an Entity Configuration (EC) at its well-known endpoint
    - The EC is valid, including:
        - It is not expired
        - It is signed with previously exchanged key
        - It contains the Trust Authority in authority_hints (alternatively, this can be done once onboarding is complete)
        - It satisfies any other requirements the Trust Authority might have regarding metadata
    - [G100.2]: EC contains specific required Trust Marks
        - Trust Marks are valid
    - Any non-technical checks
- The Trust Authority adds the Proxy to its internal database as a subordinate
- [G100.2]: Optional: the Trust Authority issues a Trust Mark to the Proxy proving membership
- [G100.2]: Optional: the Proxy adds the issued Trust Mark to its EC

To verify that the onboarding was successful:

- Go to Trust Authority’s fetch endpoint and get a statement about the Proxy
- Check that the statement is valid, i.e. not expired, contains the Proxy's public key, signed with the Trust Authority’s key
- [G100.2]: Optional: the Proxy has a valid membership Trust Mark

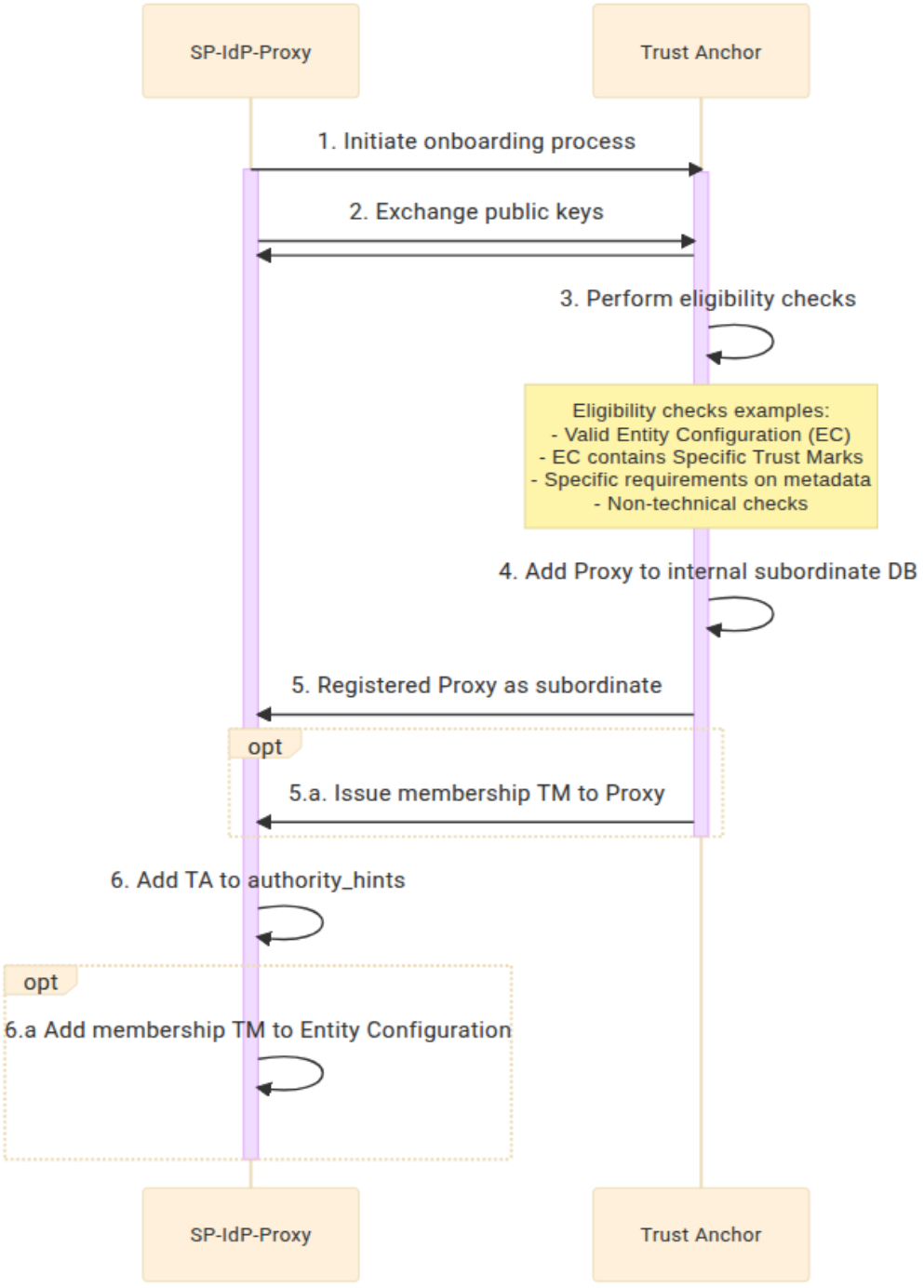


Figure 6.4: A general example for how onboarding could be implemented at a Trust Authority.

6.4.2. Entity Configuration

Each entity publishes their metadata as a signed statement called Entity Configuration in accordance to [OID-Fed]. The Entity Configuration MUST contain all the information needed for the subject entity to participate in the federation:

- The public part of the subject's signing keys
- The subject's immediate superiors
- The subject's metadata for all the roles the entity has in the federation
- [G100.2] The subject's Trust Marks

All entities in the federation MUST publish an Entity Configuration, including Trust Anchors, Intermediates, Trust Mark Issuers and SP-IdP-Proxies. SP-IdP-Proxies MUST provide OID-Fed metadata for multiple entity types, to be able to act as OIDC or OAuth 2.0 entities, such as: `openid_provider`, `openid_relying_party`, `oauth_authorization_server`, `oauth_resource`, `oauth_client`. Among these, `openid_provider` and `openid_relying_party` are mandatory in order to ensure support for standard authorization code flow, as well as AARC guidelines such as [AARC-G052] (OAuth 2.0 Proxied Token Introspection - see Chapter 7). Also, proxies MAY publish metadata for "federation_entity".

6.4.3. Resolving Trust

Figure 6.5 illustrates the general flow for how an SP-IdP-Proxy in Domain A can establish trust, acting as a Resource Server, with another SP-IdP-Proxy in Domain B acting as an OAuth 2.0 Authorization Server, both relying on a shared Trust Authority in Domain C.

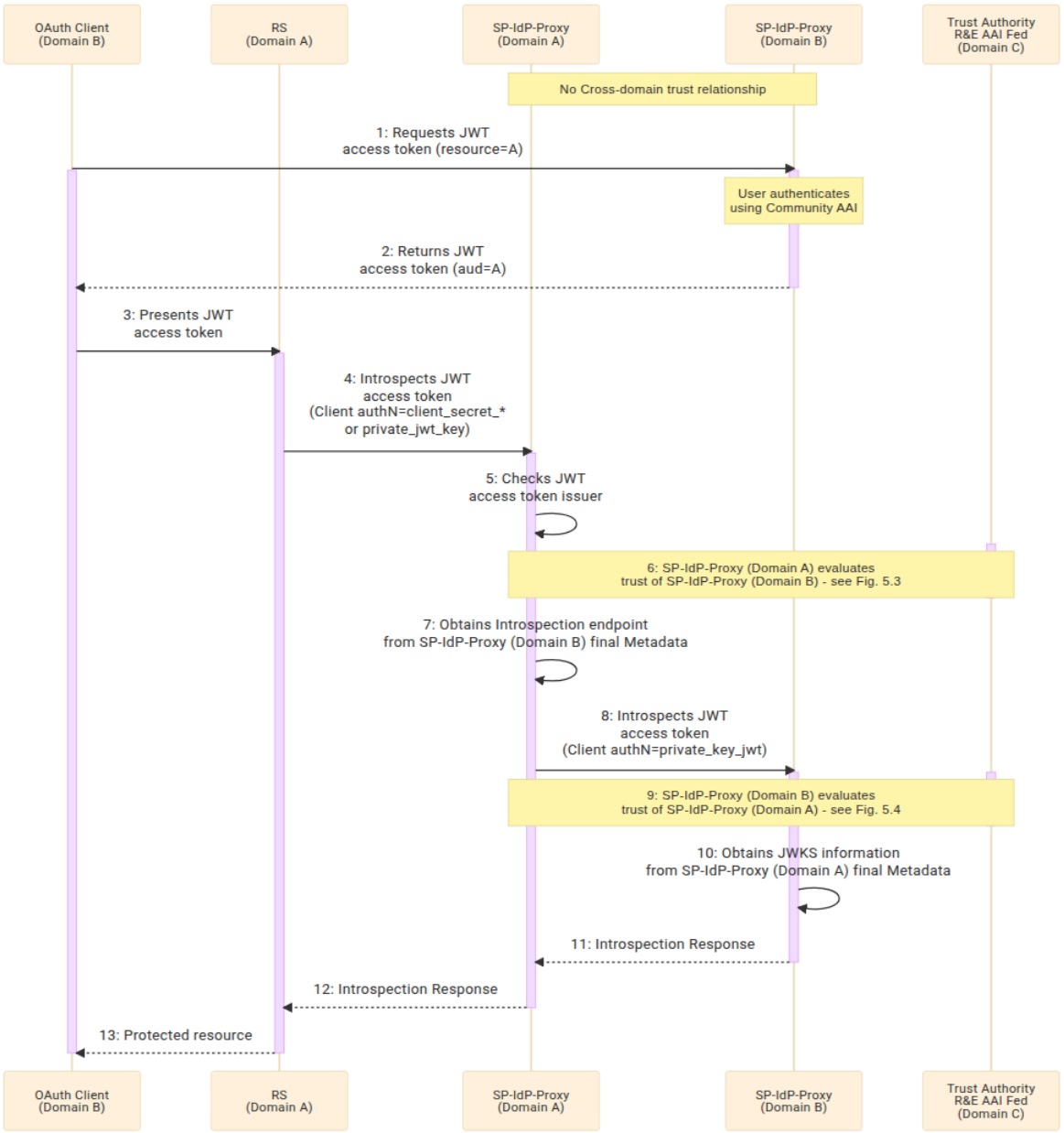


Figure 6.5: General flow for establishing trust between Proxy A acting as a Resource Server and Proxy B acting as an Authorisation Server. The flow assumes that automatic client registration is used. The diagram depicts the simple federation model with a single common Trust Authority.

In steps 6 and 9 in Fig. 6.5, each SP-IdP-Proxy retrieves the Entity Configuration from the peer entity and builds a Trust Chain that consists of multiple signed statements, starting with a statement issued and signed by the leaf entity about itself, ending with a statement issued by a Trust Anchor about itself, and all the intermediate statements being issued by a superior entity about its subordinate.

Each superior signs the public key of its subordinate, which enables a top-down verification of the entire chain. The resulting trust chain is valid if:

- It terminates at a recognised Trust Anchor.
- All intermediate statements are validly signed and unexpired.

[G100.2] To resolve trust in the fine-grained model, additional steps are required:

- As part of the Trust Chain resolution, Metadata Policies defined by each Trust Authority in the chain are applied to the subject's metadata, ensuring that technical policies are always respected.
- Required Trust Marks MUST be present and valid.

Figure 6.6 illustrates the interactions between the SP-IdP-Proxy (Domain A), the SP-IdP-Proxy (Domain B), and the Trust Authority (also a Trust Anchor in this case) during a trust evaluation made by the SP-IdP-Proxy (Domain A) for the SP-IdP-Proxy (Domain B) — Step 6 in Fig. 6.5:

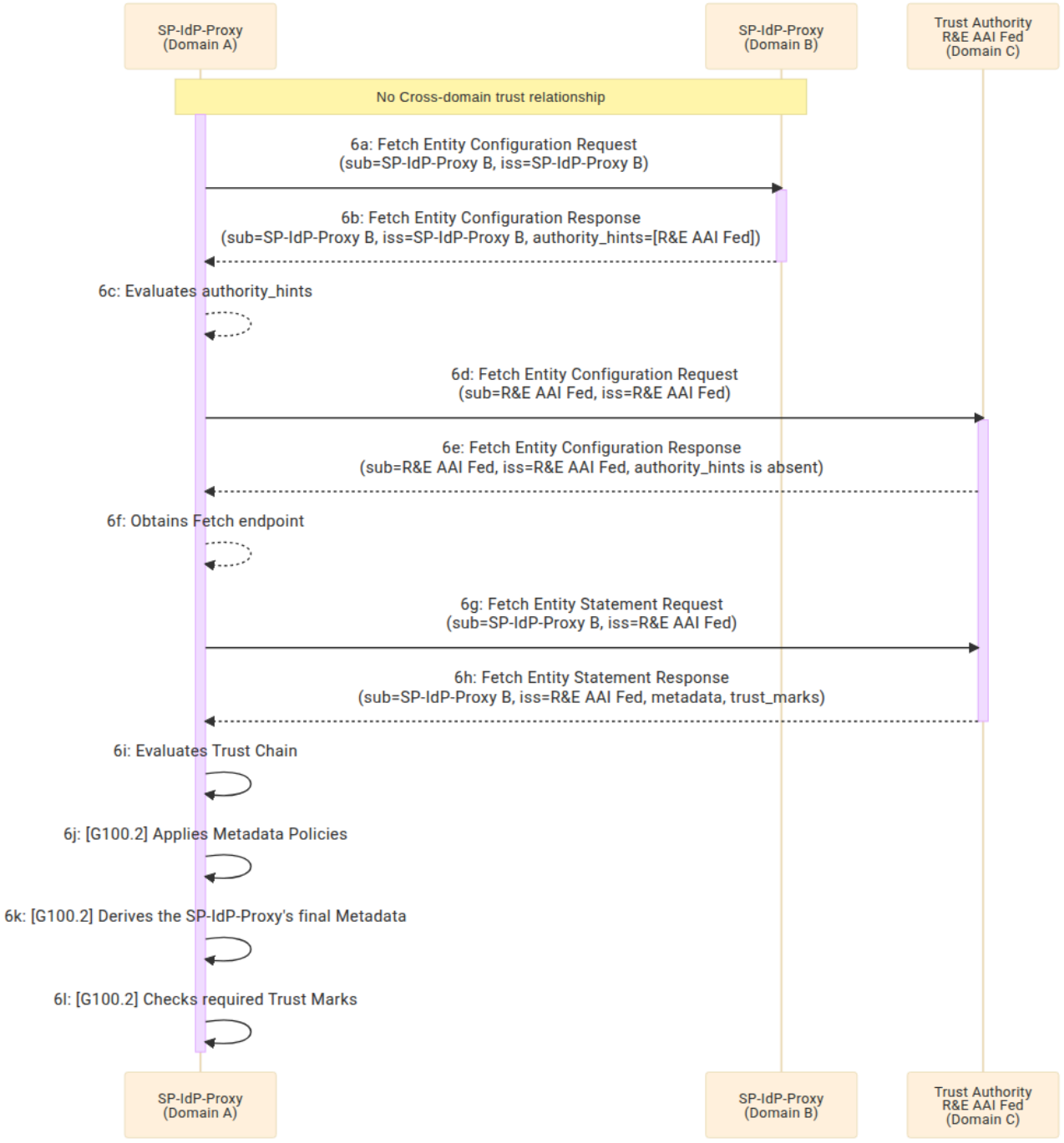


Figure 6.6: SP-IdP-Proxy (Domain A) evaluates trust of SP-IdP-Proxy (Domain B). The diagram depicts a simple federation model with a single Trust Authority. In a more complex federation with a hierarchy of Trust Authorities, steps 6c–6h are repeated until a Proxy A's Trust Anchor is reached, or until the Trust Authority does not contain any authority_hints. Multiple Trust Chains may be found, each Proxy may decide how to choose between multiple valid chains.

Figure 6.7 illustrates the interactions between the SP-IdP-Proxy (Domain A), the SP-IdP-Proxy (Domain B), and the Trust Authority during a trust evaluation made by the SP-IdP-Proxy (Domain B) for the SP-IdP-Proxy (Domain A) — Step 9 in Fig. 6.5:

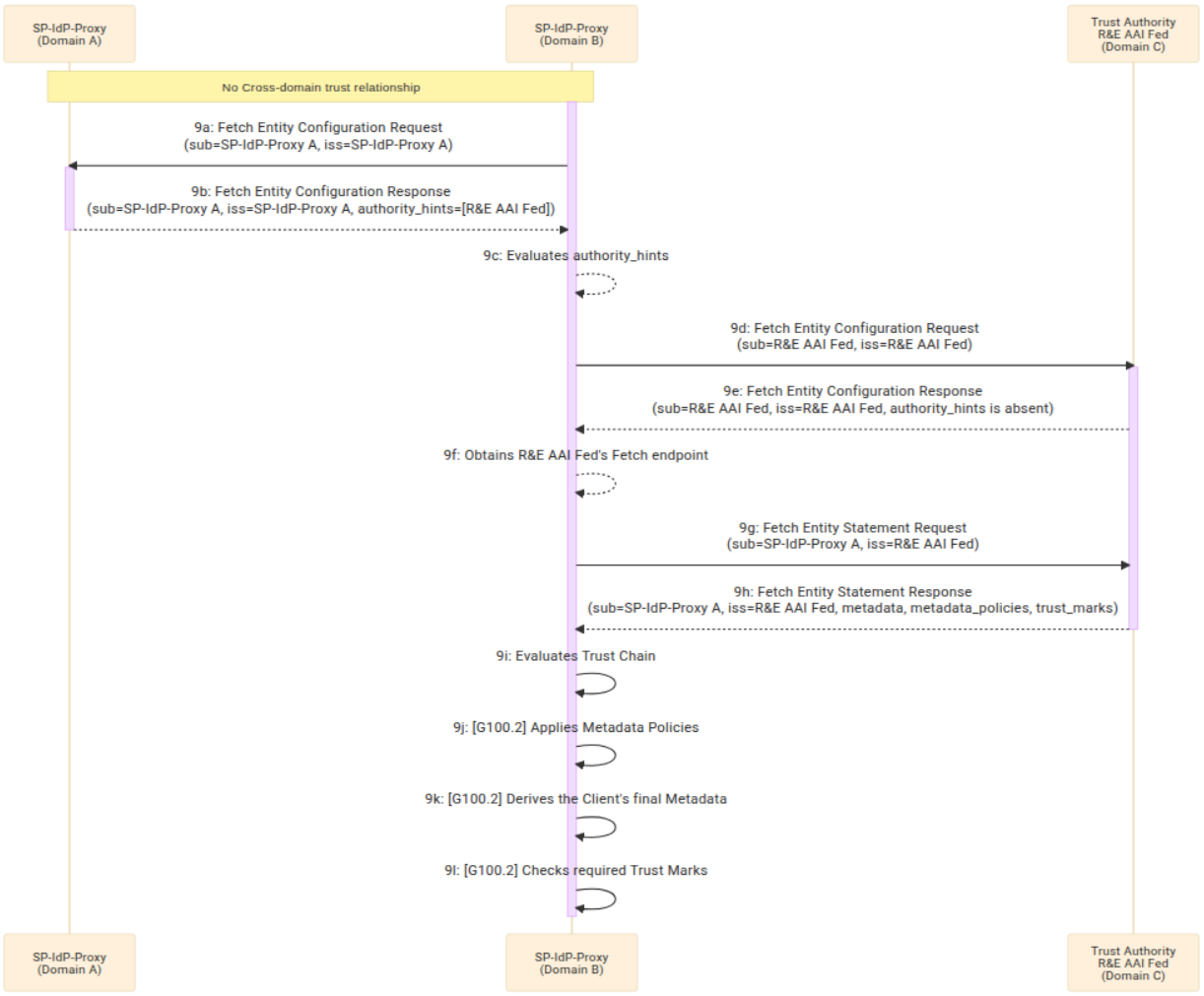


Figure 6.7: SP-IdP-Proxy (Domain B) evaluates trust of SP-IdP-Proxy (Domain A). The diagram depicts a simple federation model with a single Trust Authority. In a more complex federation with a hierarchy of Trust Authorities, steps 9c–9h are repeated until a Proxy B's Trust Anchor is reached, or until the Trust Authority does not contain any authority_hints. Multiple Trust Chains may be found, each Proxy may decide how to choose between multiple valid chains.

6.4.4. Client registration

For compliance with G100.1 and G100.2: both client registration methods MUST be supported for the OP role, one is REQUIRED for the RP role depending on the federation policies.

Automatic client registration flow:

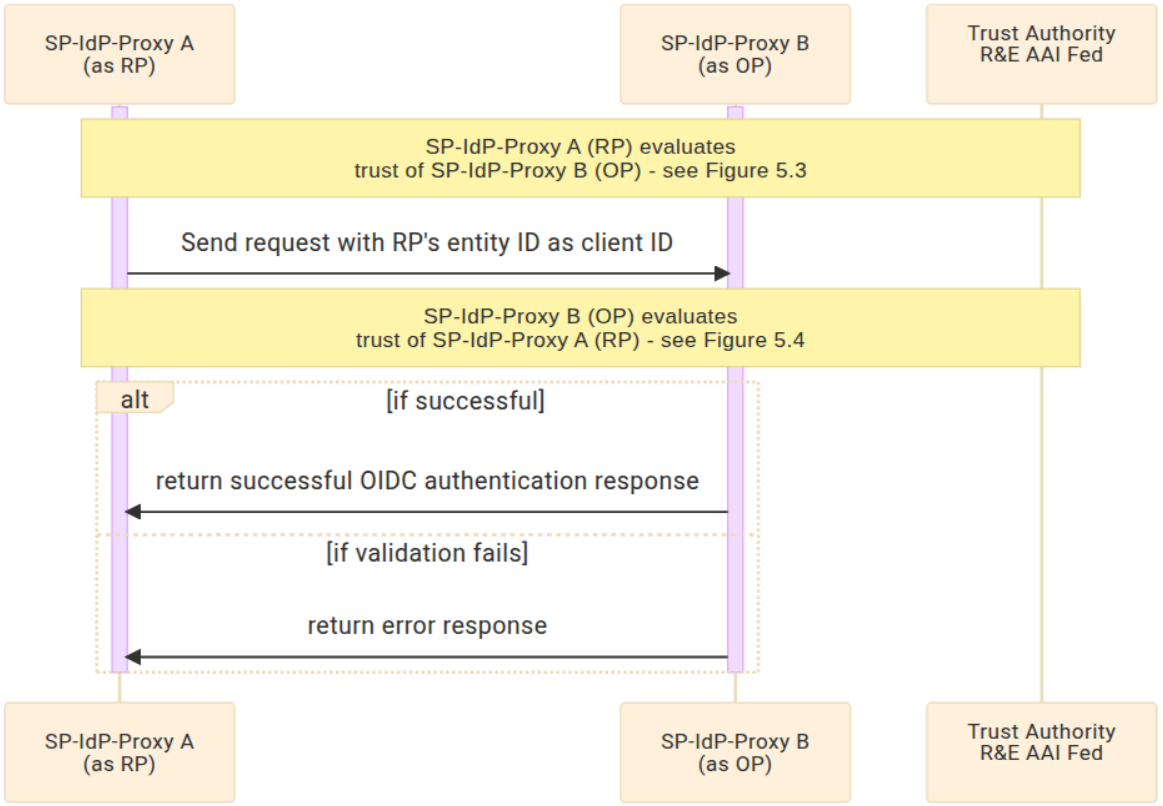


Figure 6.8: Automatic client registration

Explicit client registration flow:

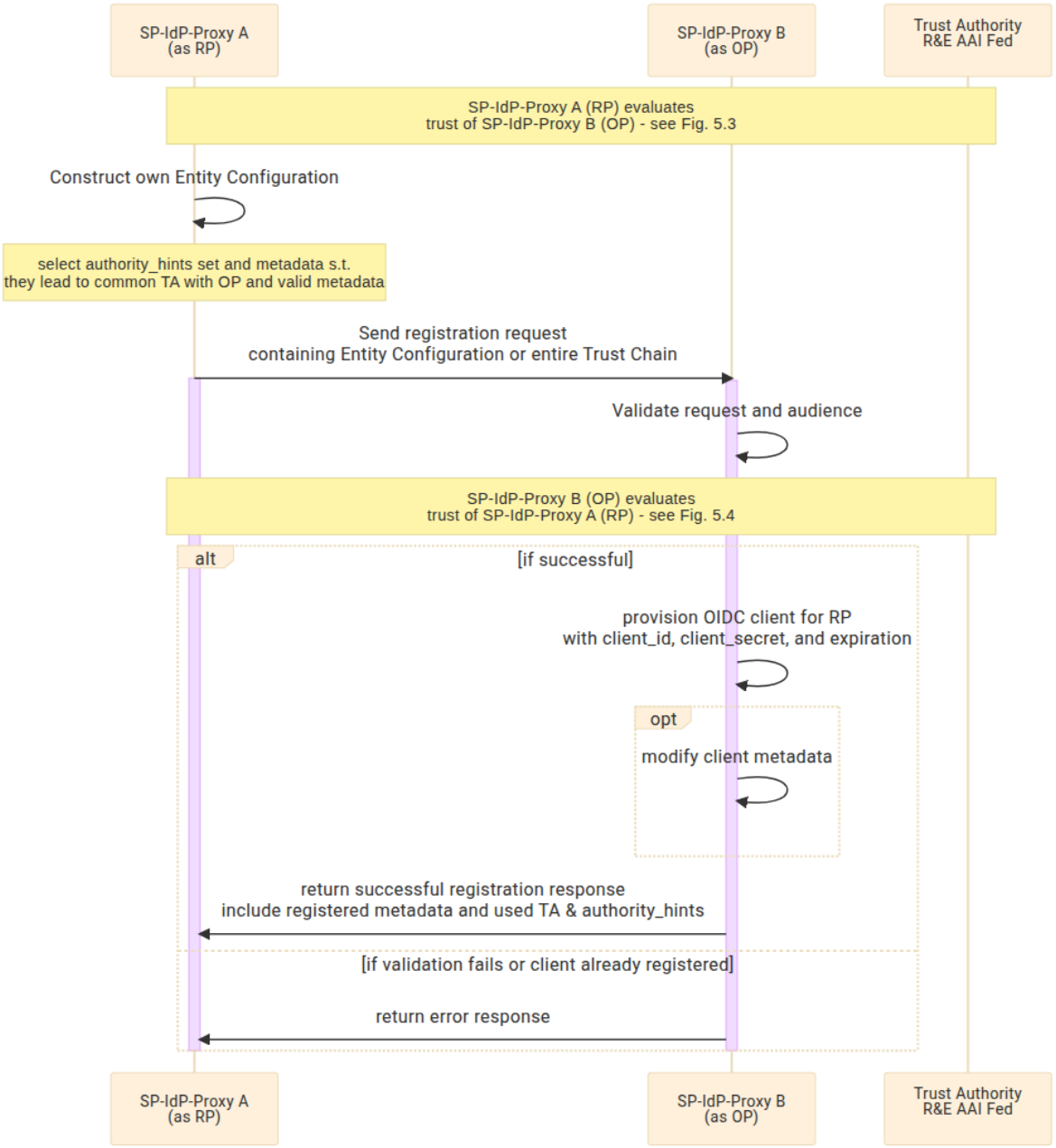


Figure 6.9: Explicit client registration

6.5. Federation Policies

In a federated AAI environment, federation policies define the expected operational, security, and organisational behaviors that participating entities must follow. These policies enable trust

interoperability across distinct trust domains, particularly when entities do not have prior bilateral agreements.

[OID-Fed] provides several mechanisms of expressing and enforcing federation policies. [G100.2] makes use of these mechanisms to enable a more fine-grained control over trust requirements:

- **Trust Marks** can be used to represent conformance to federation policies in a scalable and verifiable way.

  In addition to a Trust Mark, some federation policies might require signalling compliance as part of the transaction, via standard OIDC mechanisms, such as claims or scopes. This is the case for the REFEDS Assurance Framework [RAF], which already includes a provision on how to use the specification with OIDC. Still, its presence as a trustmark allows other entities to discover support prior to the transaction and filter based on the support for RAF. The table in Annex B.1.1 contains a non-exhaustive list of examples of federation policies that can be expressed as Trust Marks.

- **Metadata Policies** can be used to enforce requirements on the metadata of leaf entities, such as limiting supported scopes or claims, or requiring specific claims.

  Trust Authorities may use metadata policies to require Proxies to comply with specific guidelines. For example, to ensure that Proxies can release the required attributes in [AARC-G056], the metadata policy will require all OPs to support at least the following scopes via the metadata policy operator `superset_of: email, profile, entitlements`.

  An Example Subordinate Statement issued by a [G100.2] trust authority about a proxy including metadata policies is shown in Annex B.1.2.

It is also noted that federation policies still can be expressed and checked outside of OID-Fed. e.g. at the time of enrollment.

6.6. Implementation Considerations

This section is normative.

6.6.1. Configuration

6.6.1.1. Trust Authority

The Trust Authority has the following requirements:

- MUST define an onboarding process for subordinates as defined in Section 6.4.1
- SHOULD allow onboarding of entities with only a subset of the published entity types.

- MUST provide the Fetch endpoint for subordinate statements as defined in [OID-Fed]
- MUST provide the List endpoint as defined in [OID-Fed]
- [G100.2] MUST list accepted trust mark issuers
- [G100.2] MAY act as a Trust Mark issuer for the supported federation policies
- MAY provide a Resolve endpoint as defined in [OID-Fed]

An example Entity Configuration for a [G100.2] trust authority is shown in Annex B.2.1.

6.6.1.2. Trust Mark Issuer

The Trust Mark Issuer has the following requirements:

- MUST implement the Trust Mark endpoint
- MAY implement the Trust Mark Status endpoint
- MAY implement the Trust Marked Entities Listing endpoint

An example Entity Configuration for a trust mark issuer is shown in Annex B.2.2.

6.6.1.2. SP-IdP-Proxy

An SP-IdP-Proxy participating in a federation has the following requirements:

Entity Configuration Publication

- The SP-IdP-Proxy MUST expose an Entity Configuration at the well-known endpoint: `/.well-known/openid-federation` as defined in [OID-Fed].
- The Entity Configuration MUST include:
  - The Proxy's public signing keys.
  - The list of immediate superiors (authority_hints).
  - Metadata for all supported roles (see Supported Federation Roles below).
  - [G100.2] Any applicable Trust Marks.

Supported Federation Roles

- The SP-IdP-Proxy MUST publish metadata for at least the following roles:
  - `openid_provider` when acting as a Community AAI as per [AARC-G045] or Collaboration Management as per [AARC-G080]
  - `openid_relying_party` when acting as an Infrastructure Proxy per [AARC-G045] or Infrastructure Integration per [AARC-G080]
- The SP-IdP-Proxy MAY additionally publish metadata for:
  - `oauth_authorization_server` (see [AARC-G052] for additional requirements on OAuth metadata when supporting proxied token introspection)
  - `oauth_resource` (see [AARC-G052] for additional requirements on OAuth metadata when supporting proxied token introspection)
  - `oauth_client`

    - `federation_entity` when publishing generic metadata, such as informational metadata as defined in [OID-Fed] (§5.2.2)

An example Entity Configuration for a Proxy in the OP and RP role is shown in Annex B.2.3.

Trust Chain Resolution

- The SP-IdP-Proxy MUST validate peer entities as defined in [OID-Fed].
    - The SP-IdP-Proxy MAY use an external resolver service to offload trust chain construction and validation.
    - Otherwise, the SP-IdP-Proxy MUST construct and verify the Trust Chains itself. Validation MUST include:
        - Digital signature checks.
        - Expiration, subject and issuer checks.
        - [G100.2] Application of metadata policies.
        - [G100.2] Verification of required Trust Marks.
- The SP-IdP-Proxy SHOULD maintain local caching of resolved trust chains to improve performance and availability.

Client Registration

- In the OpenID Provider (OP) role, the SP-IdP-Proxy MUST implement both Automatic Registration and Explicit Registration as defined in [OID-Fed] (§12.1 and §12.2).
- The OP MUST implement the Federation Registration endpoint as defined in [OID-Fed] (§5.1.3 and §12.2)
- In the Authorization Server (AS) role, the SP-IdP-Proxy MUST implement both Automatic Registration and Explicit Registration as defined in [OID-Fed] (§12.1 and §12.2).
- The AS MUST implement the Federation Registration endpoint as defined in [OID-Fed] (§5.1.3 and §12.2)
- In the Relying Party (RP) / Client / Resource role, the SP-IdP-Proxy MUST implement at least one client registration mechanism.
- The SP-IdP-Proxy SHOULD support both registration mechanisms to maximise interoperability.
- A federation policy MAY require a certain client registration mechanism.

Trust Mark Handling [G100.2]

- The SP-IdP-Proxy MUST have the ability to validate received Trust Marks as described in [OID-Fed] (§7) during the trust resolution (see Section 6.4.3)
- The SP-IdP-Proxy MUST validate all Trust Marks that are required by federation or local policies
- The SP-IdP-Proxy MUST publish valid Trust Marks in its Entity Configuration
- The SP-IdP-Proxy MUST ensure that published Trust Marks with an expiration are refreshed and updated before they expire.

6.6.2. Federation Topologies

OpenID Federation enables a variety of federation topologies beyond the simple single Trust Authority model. Depending on deployment needs, different topologies can be combined:

- **Peer-to-peer interactions:** Direct trust establishment between proxies across administrative domains (e.g. for cross-domain resource access via proxied token introspection [AARC-G052]). Trust chains are constructed to a common Trust Anchor without requiring bilateral agreements.
- **Hierarchical federation structures:** One or more intermediate entities act on behalf of a Trust Authority, delegating trust to proxies. This topology reflects national or thematic federation operators onboarding participants under a higher-level inter-federation Trust Anchor.
- **Multiple overlapping federations:** A proxy may participate in more than one trust chain simultaneously (e.g. national, community, and cross-infrastructure federation contexts). This supports scenarios where policy requirements differ across federations, and trust decisions must consider multiple paths.

6.6.3. Performance considerations

Federation participants SHOULD consider performance aspects when implementing OpenID Federation trust resolution and client registration.

The OpenID Federation specification does not define explicit performance requirements. However, two provisions have implications for efficiency: the Resolve endpoint ([OID-Fed], Section 10.6]) and the requirement to refresh Trust Chains upon expiration ([OID-Fed], Section 11]). The Resolve endpoint allows entities to offload trust chain discovery and validation to an external resolver, which can reduce repeated computation. The refresh requirement implies the need for caching strategies, since expired Trust Chains MUST be revalidated and renewed in a timely manner.

For trust resolution, proxies acting as RPs or OPs SHOULD cache validated Trust Chains and resolved metadata until the expiration time indicated in the corresponding Entity Statements, and MAY proactively refresh chains before they expire. Where multiple valid Trust Chains exist, entities SHOULD avoid redundant processing by persisting the selected chain or prioritising Trust Anchors. For G100.2 compliance, Metadata Policies MUST be applied during Trust Chain resolution; the aggregated metadata result SHOULD be cached to avoid repeated evaluation overhead. Large-scale deployments SHOULD consider hierarchical federation structures or delegated resolvers to distribute load.

For client registration, performance depends on the chosen method. Automatic registration requires the OP to validate Trust Chains on-the-fly for every authentication request; without effective caching, this introduces latency. Explicit registration involves more up-front processing but allows reuse of the established client record, reducing runtime overhead. Since client registrations expire,

implementations SHOULD support efficient re-registration mechanisms, and OPs SHOULD optimise registration endpoints for scale. RPs MUST support reusing cached registration responses until renewal is required.

Note that, apart from Sections 10.6 and 11, the [OID-Fed] specification does not mandate specific performance behaviour. The considerations in this section are therefore implementation guidance for AARC-compliant deployments.

6.7. Security Considerations

The security considerations of [OID-Fed] (Section 18) apply.

7. Token Validation using Proxied Token Introspection

This chapter addresses cross-infrastructure validation of OAuth 2.0 access tokens using the OAuth 2.0 Proxied Token Introspection mechanism specified in [AARC-G052]. The specification extends [RFC7662] to enable the introspection endpoint of an Authorization Server (AS) that receives an introspection request for a token, not issued by itself, to initiate a new token introspection request toward a different, trusted AS to determine the validity of the set of metadata related to the token. As described in [RFC7662], the token metadata can include but are not limited to the following:

- whether or not the token is currently active (or if it has expired or otherwise become invalid),
- what access rights the token carries; rights are typically conveyed through OAuth 2.0 scopes or other authorization claims including resource owner memberships in roles and groups that are relevant to the resource being accessed, or through entitlements assigned to the resource owner for the targeted resource that the authorization server knows about (see also [RFC9068]),
- the authorization context in which the token was granted, including information such as the subject of the token and the client that the token was issued to.

Proxied Token Introspection addresses challenges in environments where there are multiple Authorization Servers (ASes) issuing tokens. In such environments, protected resources may have a trust relationship with a specific AS but may still receive OAuth 2.0 tokens issued by any of the ASes. This specification requires that each AS provide an introspection endpoint per [RFC7662] and can itself act as a client that can proxy token introspection requests and responses to another AS.

The proxied token introspection approach allows for the following:

- Resource servers can verify OAuth 2.0 tokens that were issued by ASes with which the RS is not registered; instead an AS they are registered with can proxy their token introspection requests to the AS that actually issued the token.
- If necessary, the AS performing proxied token introspection can alter, add or remove information in the token - such as authorisation statements - from the domain of the token-issuing AS to its own domain.

AARC-G052 is a finalised guideline endorsed by AEGIS. Its technical content is reused verbatim in this chapter.

7.1. Proxied Token Introspection Request and Response

The abstract flow illustrated in Fig. 7.1 describes the interactions among the entities involved in proxied token introspection. In Step 1, the OAuth 2.0 client requests an access token by

authenticating with its trusted AS, AS_2, by presenting the authorization grant. AS_2 authenticates the client and validates the authorization grant, and if valid, issues an access token.

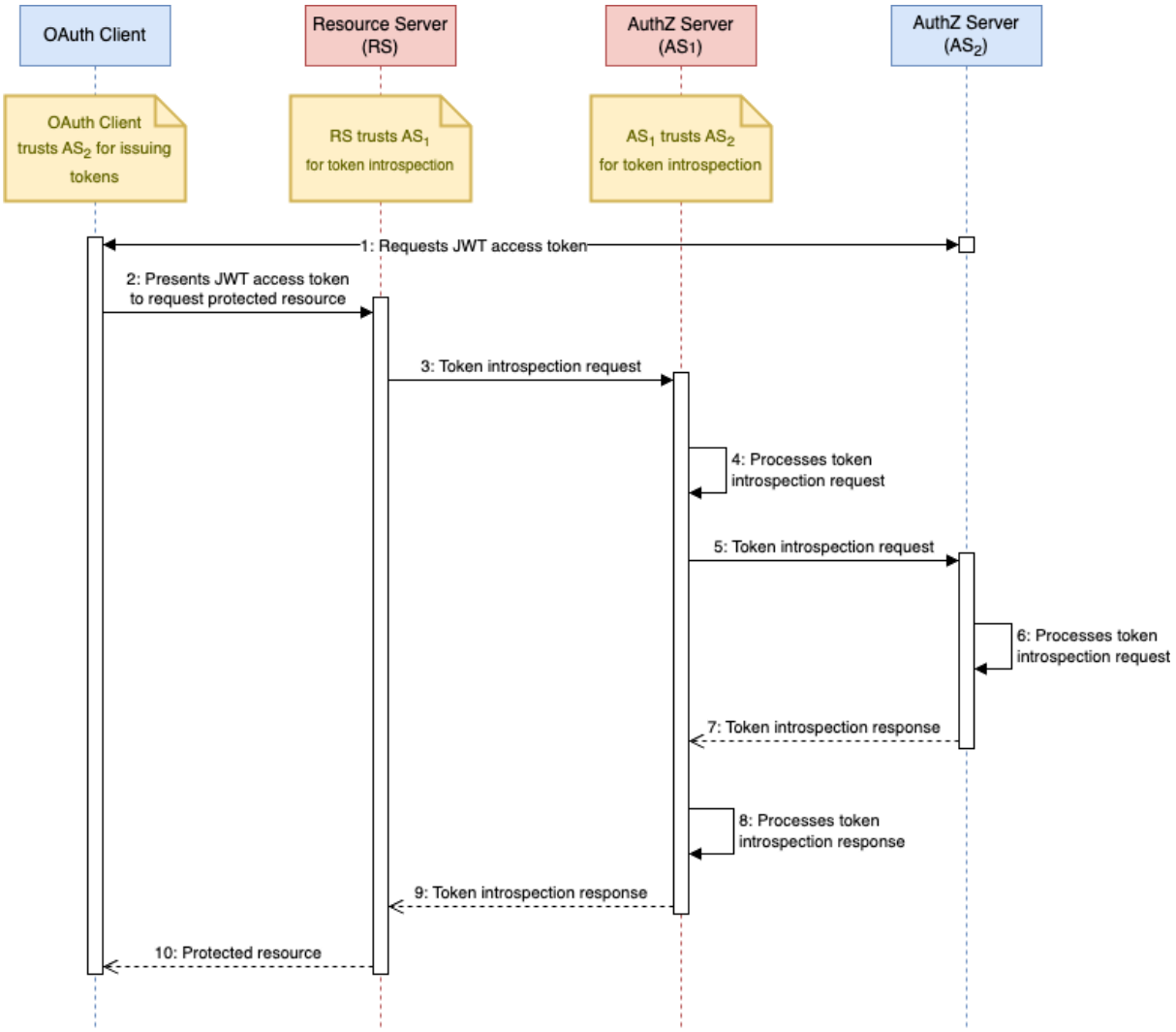


Figure 7.1: OAuth 2.0 Proxied Token Introspection flow

7.1.1. Token Validation by Resource Server

The OAuth 2.0 client uses the access token to request the protected resource from the resource server (RS) (see Step 2 in Fig. 7.1). The RS treats the token as an opaque token and calls the introspection endpoint of its trusted AS (AS_1 in Fig. 7.1) using an HTTP request as defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.1 (see Step 3 in Fig. 7.1). Access to the introspection endpoint MUST require some form of authorisation as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.1.

The following is a non-normative example request where, for the purpose of authorisation, the RS acts as an OAuth 2.0 client and uses a client identifier and client secret to authenticate itself to the introspection endpoint:

```
POST /introspect HTTP/1.1
     Host: as1.example.org
     Accept: application/json
     Content-Type: application/x-www-form-urlencoded
     Authorization: Basic cnM6Z3FWWFFUZmVMSHRTdnhzTg==

     token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
     ovL2FzMi5leGFtcGxlLm9yZy8iLCJzdWIiOiI1YmE1NTJkNjciLCJleHAiOjE
     3NDYwODMxNjYsImlhdCI6MTc0NjA3OTU2NiwianRpIjoiZGJlMzliZjNhM2Jh
     NDIzOGE1MTNmNTFkNmUxNjkxYzQiLCJjbGllbnRfaWQiOiJzNkJoZFJrcXQzI
     iwic2NvcGUiOiJvcGVuaWQgcHJvZmlsZSBlbWFpbCBlbnRpdGxlbWVudHMifQ
     .K9TnkoCRjYbCDAvvIDMIr4O0SPeK8B22Pd2xKAZgxaA
```

The token in the example includes the following claims:

```
{
  "iss": "https://as2.example.org/",
  "sub": "5ba552d67",
  "exp": 1746083166,
  "iat": 1746079566,
  "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile email entitlements"
}
```

7.1.2. Token Validation by AS Performing Proxied Token Introspection

In processing the request, AS_1 MUST determine whether the token has been issued by itself (see Step 4 in Fig. 7.1). This specification requires signed access tokens to be presented in JWT format containing the iss claim for identifying the issuer of the token, as defined in JWT [RFC7519]. The handling of opaque and/or encrypted access tokens is out of scope of this specification (see also Sections 7.2 & 7.4).

If AS_1 determines that the token has been issued by itself, then an introspection response as defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.2, MUST be returned.

If the token has not been issued by AS1 itself, it MUST either:

a. perform Proxied Token Introspection, by acting as a resource server and calling the introspection endpoint of another AS it has a trust relationship with (AS_2 in Fig. 7.1), as specified in [RFC7662], Section 2.1, or

b. return an introspection response with the "active" field set to "false", as specified in [RFC7662], Section 2.2 (see Section 7.1.4).

The means by which the trusted AS is selected is out of the scope of this specification. For example the AS may be selected based on the value of the iss claim found in the access token or statically as a pre-configured trusted AS for the purpose of (again) performing Proxied Token Introspection. Similarly, how AS_1, acting as an OAuth 2.0 resource server, discovers the location of the introspection endpoint of the trusted AS is outside the scope of this specification; for instance, AS_1 may use the OAuth 2.0 Authorization Server Metadata [RFC8414], Section 2 to discover the location of the introspection endpoint. The AS - acting as an OAuth 2.0 client itself - calls the introspection endpoint of the trusted AS (AS_2 in Fig. 7.1) using an HTTP request as defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.1 (see Step 5 in Fig. 7.1). The introspection endpoint of AS_2 MUST support client authentication as described in OAuth 2.0 [RFC6749] to authorise introspection requests from AS_1. The means by which an AS discovers the client authentication methods supported by the introspection endpoint of another AS are outside the scope of this specification; for instance, AS_1 may use the OAuth 2.0 Authorization Server Metadata [RFC8414], Section 2 to discover the supported client authentication methods.

The following is a non-normative example request where AS_1 as an OAuth 2.0 client uses a client identifier and client secret to authenticate itself to the introspection endpoint of another trusted AS:

```
POST /introspect HTTP/1.1
     Host: as2.example.org
     Accept: application/json
     Content-Type: application/x-www-form-urlencoded
     Authorization: Basic YXMxOkFkWmhlS3E0N0ttMnJiWkw=

     token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwcz
     ovL2FzMi5leGFtcGxlLm9yZy8iLCJzdWIiOiI1YmE1NTJkNjciLCJleHAiOjE
     3NDYwODMxNjYsImlhdCI6MTc0NjA3OTU2NiwianRpIjoiZGJlMzliZjNhM2Jh
     NDIzOGE1MTNmNTFkNmUxNjkxYzQiLCJjbGllbnRfaWQiOiJzNkJoZFJrcXQzI
     iwic2NvcGUiOiJvcGVuaWQgcHJvZmlsZSBlbWFpbCBlbnRpdGxlbWVudHMifQ
     .K9TnkoCRjYbCDAvvIDMIr4O0SPeK8B22Pd2xKAZgxaA
```

7.1.3. Token Introspection Response by the token-issuing AS

The token-issuing AS (AS_2 in Fig. 7.1) responds to the AS performing the Proxied Token Introspection (AS_1 in Fig. 7.1) with a JSON object [RFC7159] in "application/json" format with the top-level members defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.2 (see Step 7 in Fig. 7.1).

7.1.4. Token Introspection Response by AS performing Proxied Token Introspection

The AS performing the Proxied Token Introspection (AS_1 in Fig. 7.1) responds to the resource server with a JSON object [RFC7159] in `"application/json"` format with the top-level members defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.2 (see Step 9 in Fig. 7.1). The response SHOULD include the claim values included in the response from the token-issuing AS (AS_2 in Fig. 7.1). AS_1 MAY adjust the claims and/or claim values included in the response considering the requirements of the resource server which made the original request (see Step 8 in Fig. 7.1).

The following is a non-normative example of a response for an active token:

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

{
  "active": true,
  "iss": "https://as2.example.org/",
  "sub": "5ba552d67",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile email entitlements",
  "exp": 1746083166,
  "iat": 1746079566,
  "entitlements": [
    "e9e30dba-f08f-4109-8486-d5c6a331660a",
    "entitlement2"
  ]
}
```

The AS performing the Proxied Token Introspection (AS_1 in Fig. 7.1) MUST return an introspection response with the `"active"` field set to `"false"` if the introspection call is properly authorised but any of the following conditions apply:

- this particular token is not valid for use by the RS making the request[3]; for instance, if AS_1 determines that the token is not locally issued and is of an OAuth 2.0 token type [OAuth-TT] which cannot be used as an OAuth 2.0 bearer token, such as a refresh token,
- AS_1 can not validate the token through any of the trusted AS (refer to Section 7.1.2).

The following is a non-normative example response for a token that has been revoked or is otherwise invalid:

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

[3] The details of how the AS makes such a decision are outside the scope of the AARC-G052 specification.

```
{
  "active": false
}
```

If the introspection call is not authorised then the AS performing the proxied token introspection MUST respond with an HTTP `401` code as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.3.

7.2. Implementation Considerations

The AS is an entity that implements Proxied Token Introspection as defined in this specification. The AS may be a system of one or more ASes acting as a frontend for one or more other ASes. Although in the examples of this document we present a topology where the AS trusted by the RS has a direct trust relationship with the AS that has issued the OAuth 2.0 Token, more complex topologies are also possible where between the AS of the RS and the token-issuing AS there is chain of ASes that implement the Proxied Token Introspection as described in this document.

The means by which the RS chooses which entity to invoke are outside the scope for this specification. Note that this is analogous to the discovery of the token introspection defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.

When forming its proxied token introspection response, the AS MUST NOT change claims related to the token itself (i.e. the `iss`, `exp`, `iat`, `nbf`, `token_type`, `client_id`, and `jti` claims value of the response), but it MAY modify other claims or their values. Furthermore, the AS MAY respond differently to different resource servers making the same request, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2. For instance, the AS implementing this specification MAY limit which scopes (from a given token) are returned to each resource server. In other cases, the AS implementing this specification MAY limit, extend or modify the entitlements [RFC9068] from a given token depending on the protected resource making the request. Additionally, the AS MAY change the string identifier(s) representing the intended audience for the token.

The response to a properly formed and authorised query for an inactive or otherwise invalid token is not considered an error response by this specification. Instead, the AS MUST respond with an introspection response with the `"active"` field set to `"false"`. That response SHOULD NOT include any additional information about the inactive token, including why the token is inactive, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2.

As described in OAuth 2.0 Token Introspection [RFC7662], Section 4, the response MAY be cached by the resource server to improve performance, based on its own policy and risk profile. In the context of this specification, the AS performing Proxied Token Introspection (AS_1 in Fig. 7.1) MAY also cache responses from the trusted AS (AS_2 in Fig. 7.1) to reduce load on each of the ASes that are in the path between the AS trusted by the RS and the token-issuing AS. Such caching introduces a tradeoff

between performance and the freshness of token validation information, as AS_1 may not perform real-time validation for each request, which contrasts with the expectations of [RFC7662] for up-to-date checks. To mitigate risks associated with cached responses, implementers SHOULD consider countermeasures, such as mechanisms to ensure transparency and freshness of cached data, though such measures are outside the scope of this specification and may be defined in specific profile extensions to AARC-G052. Implementers should evaluate whether caching is suitable for their deployment, considering the RS's reliance on fresh token state information.

The token-issuing AS MAY respond differently depending on the identity of the entity performing the token introspection request, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2. In addition, some AS implementations enforce audience restrictions by requiring that the requestor is included in the audience for the token. While such checks are important for securing token use, the specification of how to signal or validate audiences is out of scope of both [RFC7662] and this specification. Profiles building on this specification SHOULD define mechanisms for audience signalling and validation (e.g. using audience claims or resource indicators [RFC8707]) to ensure that tokens are used only by their intended recipients.

7.3. Security Considerations

The security considerations discussed in OAuth 2.0 Token Introspection [RFC7662], Section 4 also apply to this specification. In addition to these considerations, this specification requires mechanisms (such as [RFC7515]) to ensure that access tokens presented in JWT format get delivered without having been tampered with. Although the token-issuing AS can use any algorithm defined in [RFC7518, RFC9864] for signing JWT access tokens, use of asymmetric keys is RECOMMENDED as it simplifies the process of acquiring validation information. As per [RFC9068], Section 2.1, signed JWT access tokens MUST NOT use "none" as the signing algorithm. Furthermore, authorization servers conforming to this specification MUST include RS256 (as defined in [RFC7518], Section 3.1) among their supported signature algorithms. Entities performing proxied token introspection are not required to validate the signature of the token.

As per RFC7662, Section 4, if the token can be used only at certain resource servers, the AS MUST determine whether or not the token can be used at the entity making the introspection call. How the client or the AS issuing the token is able to specify the correct audience(s) is out of scope of this specification. As described in Section 7.2, the AS MAY adjust the introspection response contents based on the resource server making the request.

For cross-infrastructure use of access tokens, the protected resource treats the access token as opaque and hence there is no risk of Cross-JWT Confusion ([RFC8725], Section 2.8).

The following mechanisms can prevent token scanning attacks and overloading of the token introspection endpoint:

- the AS implementing this specification MUST require some form of authorisation to the introspection endpoint, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.1,
- the AS implementing this specification SHOULD employ rate-limiting mechanisms ([RFC7662], Section 4), based on the identity of the RS performing the token introspection request
- the RS MAY cache responses, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2.
- the AS implementing this specification MAY cache responses, as described in Section 7.2.

7.4. Privacy Considerations

The use of Proxied Token Introspection introduces constraints on the use of encrypted access tokens, such as JSON Web Encryption (JWE). Since ASes cannot inspect the contents of encrypted tokens (e.g. the `iss` claim) without access to decryption keys, they cannot reliably make routing or delegation decisions. Consequently, access tokens intended for use across trust boundaries—where resource servers rely on an AS different from the token-issuing AS—must remain inspectable by the AS performing Proxied Token Introspection. This requirement implies that full encryption of token contents is not feasible in such scenarios. This constraint represents a privacy tradeoff, as access tokens must remain unencrypted to support interoperability in federated environments.

The content of the JWT access token will be accessible to the resource server and eventually to the AS that performs proxied token introspection. Therefore, measures MUST be taken to prevent disclosure of personal data in the JWT claims to unintended parties. For instance, all sensitive data could be left out of the token and be released only in an introspection response. In general, measures MUST be taken to prevent disclosure of this information to unintended parties., as discussed in OAuth 2.0 Token Introspection [RFC7662], Section 5.

8. Verifiable Credentials and Digital Identity Wallets

This chapter examines the relationship between the AARC Blueprint Architecture and emerging decentralised identity technologies, with a focus on Verifiable Credentials and Digital Identity Wallets.

The chapter is based on the informational document Verifiable Credentials and Digital Identity Wallets in Research Collaborations (AARC-I101). Relevant technical and architectural content from AARC-I101 is reused verbatim.

AARC-I101 is a non-normative informational document. It does not define mandatory requirements for AARC-compliant AAIs, but analyses architectural patterns for wallet-based credential models in federated research environments. The document has been shared with the AEGIS group for discussion and is intended to inform future guidelines, pilots, and standardisation activities.

8.1. Overview of Concepts and Specifications

This section summarises the core concepts of Verifiable Credentials and Verifiable Presentations. Detailed definitions are provided in the W3C Verifiable Credentials Data Model [W3C-VC-DM2] and relevant OpenID specifications [OID4VCI, OID4VP].

8.1.1. Core Concepts

A **Verifiable Credential (VC)** is a tamper-evident assertion containing a set of one or more claims about a subject, cryptographically signed by an Issuer.

A **Verifiable Presentation (VP)** is a cryptographically verifiable artefact used by a Holder to present one or more Verifiable Credentials, or selected claims from them, to a Verifier, typically for a specific purpose or transaction.

The **Holder** is the entity that receives and controls Verifiable Credentials and initiates Verifiable Presentations. A **Digital Identity Wallet** allows the Holder to manage such credentials and selectively disclose information to a Verifier.

As illustrated in Fig. 8.1, the trust relationship among **Issuer**, **Holder** (via their **Wallet**), and **Verifier** requires trust not only between the issuing and verifying services but also on the Holder's side. The Holder must trust the Issuer to provide accurate attestations and the Verifier to handle the presented data correctly. Equally important, the Holder must trust their Digital Identity Wallet and its underlying device for secure, user-controlled storage and key management.

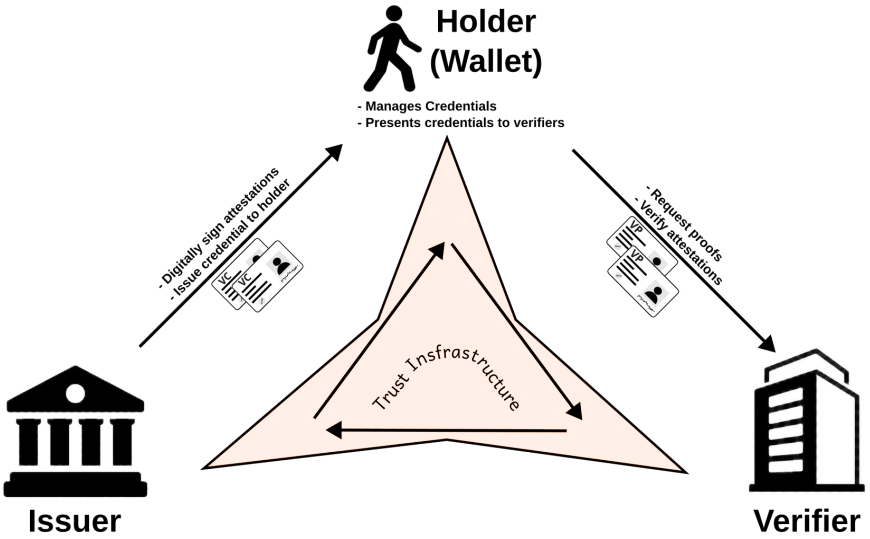


Figure 8.1: The VC trust triangle: Issuers provide signed VCs to the holder's wallet. The holder can present a selectable subset (VPs) to Verifiers. Trust is established by: (a) Verifiers needing to trust Issuers. (b) Issuers needing to trust Wallets. (c) Holders/Wallets needing to trust Verifiers to handle attributes correctly. The Holders must also trust the Issuer (to provide accurate data) and their own Wallet (device) to securely manage their keys.

To understand how these components operate, it is necessary to distinguish between the static assertions and the dynamic protocols used to share them. Verifiable Credentials and Verifiable Presentations serve distinct architectural purposes:

- Verifiable Credentials (VCs) are cryptographically verifiable artefacts issued by authoritative entities (Issuers) under a defined governance and trust framework. They encode assertions about a subject and can be stored and reused beyond a single interaction.
- Verifiable Presentations (VPs) are runtime interactions through which a Holder selectively discloses one or more credentials to a Verifier, possibly applying privacy-enhancing techniques such as selective disclosure or zero-knowledge proofs.

In other words, VC represents governed data, while VPs represent governed interactions. This distinction is essential when later mapping VC/VP usage onto the AARC Blueprint Architecture.

8.1.2. Relevant Specifications

Area	Standard / Initiative	Scope
Core Model	W3C VC Data Model 2.0 [W3C-VC-DM2]	Defines credential structure and verification semantics
Protocols	OpenID for Verifiable Credential Issuance [OID4VCI], OpenID for Verifiable Presentations [OID4VP], Selective Disclosure for JSON Web Tokens (SD-JWT) [RFC9901], OpenID Federation for Wallet Architectures [OID-Fed-Wallet]	Enable issuance and presentation flows using OIDC paradigms
Ecosystem	EUDI Wallet Architecture and Reference Framework [EUDI-ARF]	Specifies architecture and governance for EU wallets

8.2. Architectural Considerations for Wallet-based Credential Flows in the AARC Blueprint Architecture

This section describes how Verifiable Credentials (VCs), Verifiable Presentations (VPs), and Digital Identity Wallets could be used in AARC-compliant AAIs, examining architectural patterns, message flows, and component implications.

VCs introduce new participants (Issuer, Holder, Verifier) and new interaction models based on cryptographic credential issuance and presentation, which can complement the federated Single Sign-On (SSO) and attribute release models commonly used in research collaborations, extending them beyond session-bound assertions.

Trust in VC-based interactions continues to rely on governed trust frameworks, organisational accountability and established federation principles (for details see Section 8.3).

Several options exist for mapping these new participants and interactions onto existing components (Home IdPs, Community AAIs, Infrastructure Proxies, Attribute Authorities, Services). These are discussed in this section, with particular attention to where credential verification and policy enforcement take place.

8.2.1. Architectural Patterns

This section explores different patterns for the use of Verifiable Credentials and Verifiable Presentations within existing research Authentication and Authorisation Infrastructures. These

patterns describe how the roles of Issuer, Holder and Verifier may be mapped onto existing BPA components and Home Identity Providers and Services.

The following principles are common to all patterns described below:

- Verifiable Credentials are issued by entities operating within established Trust Frameworks.
- Verifiable Presentations are created by user-controlled Wallets and represent runtime interactions.
- Infrastructure Proxies or Community AAIs act as the Verifiers of Verifiable Presentations in BPA-aligned architectures.
- Community AAIs act as the Issuers of Verifiable Credentials
- Services consume authorisation context, either directly as Verifiable Presentations or in a form derived from them.

In the baseline scenario, the user authenticates locally (via a wallet) and presents a VP to a component of the research infrastructure — in the AARC context this is typically the Proxy. The Proxy validates the credential, extracts the necessary attributes, and follows established mechanisms to issue a trusted assertion for downstream services. This pattern mirrors flows already familiar from token translation or attribute aggregation.

With Verifiable Credentials, the range of Issuers is broader. Users are expected to collect credentials from multiple Issuers in their Wallets, allowing the Proxy to obtain attestations about a user's identity (e.g. government-issued identity), in addition to attestations about their institutional affiliation and associated roles (e.g. as in the eduPerson schema [EDUPERSON]), as well as identity assurance (e.g. as in the REFEDS Assurance Profile [RAF]).

Figure 8.2 provides an overview of the architectural patterns discussed in this section by illustrating how the Issuer and Verifier roles can be realised within the AARC Blueprint Architecture. In particular, it shows that AARC-BPA Proxies may assume the role of an Issuer and/or a Verifier. The following subsections examine these patterns in more detail, describing how Verifiable Credential issuance and presentation are mapped onto existing BPA components and interaction models.

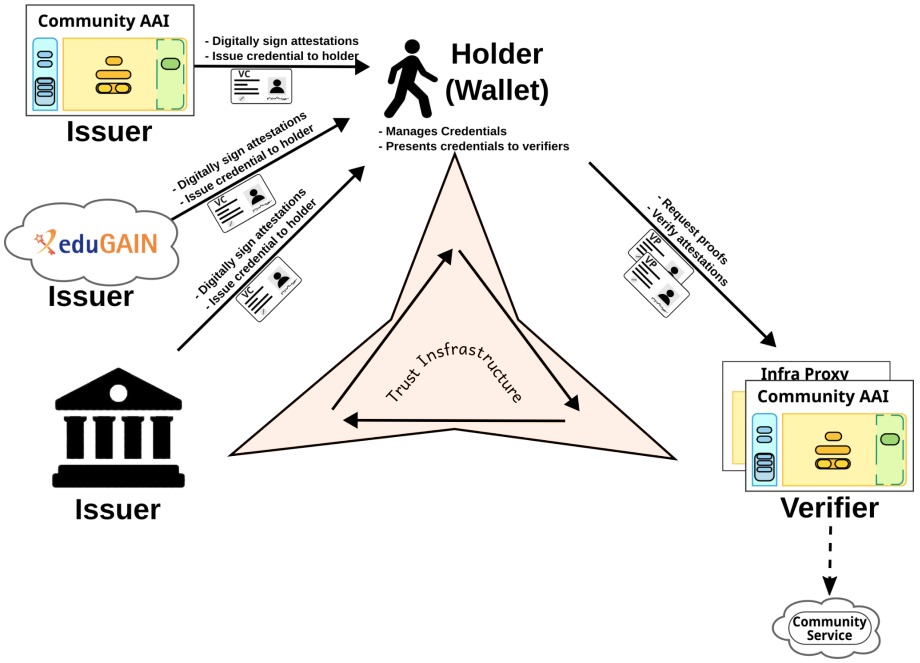


Figure 8.2: Possible roles of AARC-BPA Proxies in Verifiable Credential-based flows.

8.2.1.1. Credential Issuers

Following the EUDI concept, the user's Home Organisation (e.g. a university or research institute), as well as the Community AAI can act as Credential Issuers. Multiple issuers may issue VCs to a Holder's Wallet. Issuers we consider in addition to governmental ones, that provide well-assured identity information (i.e. RAF-IAP high), are the institutional home-IdPs, as well as AARC Community AAIs. The Home IdPs are authoritative sources for information (such as [SCHAC] and [EDUPERSON]) and potentially "VC information" (such as diploma, and courses). The AARC Community AAIs provide additional VCs such as collaboration-specific attributes (entitlements, memberships, roles).

This approach preserves the BPA principle that services remain isolated from identity and trust complexity, while enabling Home-issued credentials to be reused across collaborations. The combination with government-issued VCs promises a straightforward support for the Identity-Assurance of the REFEDS Assurance Framework [RAF].

A Community AAI that issues verifiable credentials to the user's wallet may encode:

- project affiliation, collaboration group membership and role information,
- resource Capabilities,
- training completion, or
- other structured collaboration attributes.

The issuance path allows users to carry research-specific identity information into other contexts or reuse them across infrastructures.

8.2.1.2. Credential Verifiers

As in other BPA-aligned patterns, the preferred integration model positions the Infrastructure Proxy as the Verifier of Verifiable Presentations, ensuring consistent policy evaluation across services. In this scenario, Verifiable Presentations are created by the user's Wallet at runtime, based on credentials issued by the Home Organisation. These presentations are not consumed directly by services, but are instead presented to a Community AAI or an Infrastructure Proxy, which assumes the role of Verifier. The Proxy validates the presentation, evaluates it against federation and community policy, and translates the outcome into standard authorisation assertions consumable by downstream services.

Direct verification of Verifiable Presentations by services remains possible but introduces additional complexity and governance responsibilities. Such patterns should therefore be considered advanced and context-specific. When services assume the Verifier role, they must also take responsibility for trust framework alignment, policy enforcement and auditability.

8.2.1.3. Wallets

As shown in Fig. 8.2, no AARC Blueprint Architecture component directly maps to the Digital Identity Wallet role. The wallet is a conceptually distinct entity, designed to operate under direct user control — typically on personal devices such as smartphones or browsers — and decoupled from issuers and verifiers.

Although some wallet functions overlap with existing Community AAIs (e.g. attribute aggregation) and others could be added (e.g. selective disclosure), full wallet implementations require secure key management, cross-platform compatibility, sophisticated user interfaces, and robust interaction with multiple issuers and verifiers.

Community AAIs and Infrastructure Proxies remain well-suited to act as Issuers or Verifiers, issuing research-specific credentials to user wallets or validating presentations from them. Both issuance and verification paths can coexist within the same architecture.

8.2.2. Architectural Flows

The SP-IdP-Proxy (or Proxy) defined in AARC-BPA-2019 (AARC-G045) can operate as an Issuer or Verifier for Wallet-presented credentials. This aligns well with its existing architectural responsibilities: aggregating attributes, harmonising trust, and providing a consistent abstraction to downstream services.

A typical sequence for the patterns discussed in Section 8.2.1 involves the following steps:

1. The Issuer issues a VC to a Holder, who stores it in the wallet.
2. The Proxy (acting as a Verifier) requests the Holder to identify / provide a set of claims.
3. The Holder retrieves a VP from their wallet.

4. The Proxy verifies the credential's signature, issuer metadata, and validity.
5. The Proxy normalises the attributes (e.g., mapping JSON-LD or SD-JWT claims to internal attribute schemas).
6. The Proxy (acting as an OIDC OP or Oauth2.0 AS) issues a new JWT/OIDC token trusted by services.
7. Downstream services consume the Proxy-issued token without needing to understand the credential format or the issuer ecosystem.

This Proxy-mediated trust delegation allows relying parties to trust one entity (the Proxy) rather than every external VC issuer.

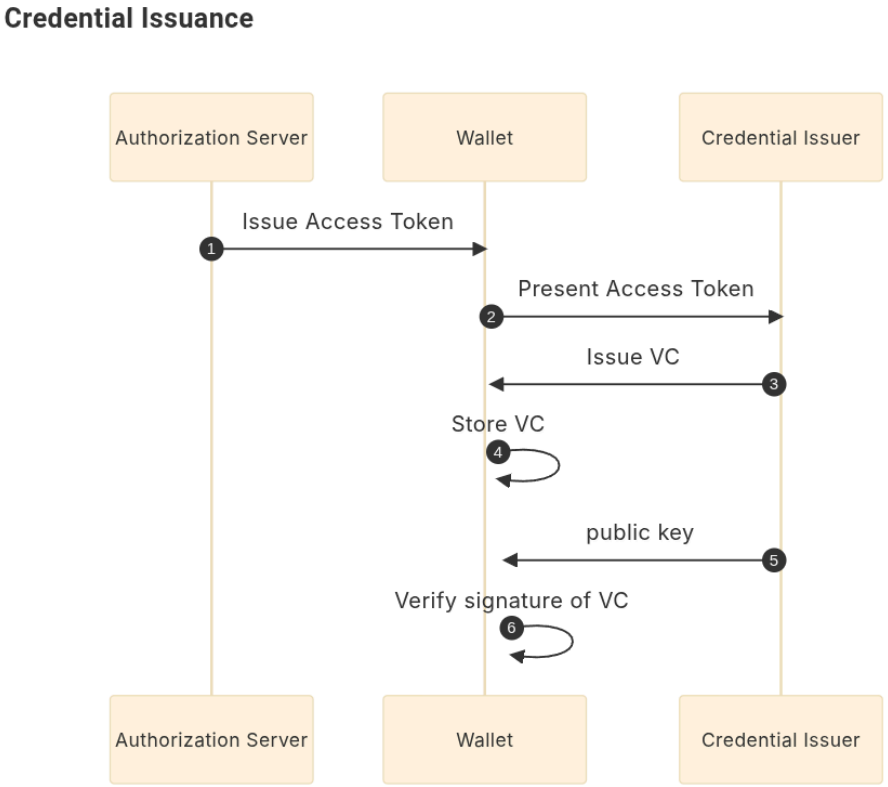


Figure 8.3: Sequence for obtaining a Verifiable Credential

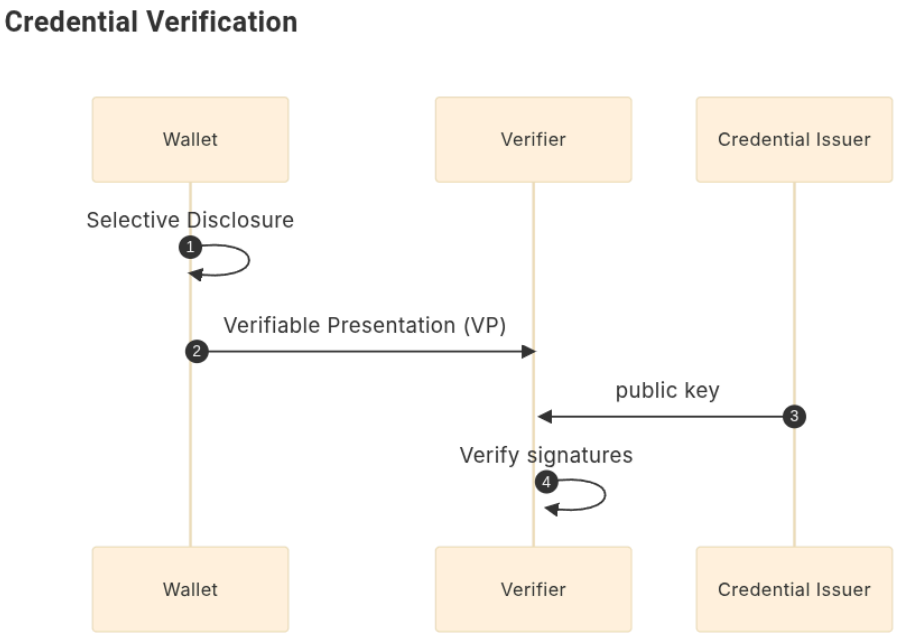


Figure 8.4: Sequence for Selective Disclosure and Credential Verification

8.2.3. Component-level Requirements for the BPA

The following subsections outline the specific requirements and differences introduced when AARC Blueprint Architecture components assume the roles of Credential Issuer, Verifier, or interact with Wallets.

8.2.3.1. Issuer Requirements

Community AAIs (and Identity Proxies) can extend their existing OpenID Provider (OP) functionality to support Verifiable Credential issuance in accordance with [OID4VCI]). A fundamental distinction in this model is the provision of persistent, portable credentials, as opposed to the session-based tokens of standard OIDC.

To implement this capability, the following requirements apply:

- **Multiple Format & Schema Support** - Must handle JWT-VC, JSON-LD, SD-JWT formats with structured credential templates vs only
- JWT
    - Support for W3C VC Data Model 2.0 and JSON-LD contexts
    - Predefined credential schemas for specific use cases (education, employment, etc.)
    - Template-based credential generation with standardized claim structures
- **Deferred Issuance** - Async credential generation vs immediate token issuance
    - Support for approval workflows and manual verification processes
    - Polling mechanisms for credential readiness status

    - Notification systems for credential availability
- **Revocation Management** - Maintain revocation lists vs simple token expiration
    - Real-time revocation list publication and updates
    - Integration with revocation registries and status checking
    - Credential lifecycle management beyond simple expiration
- **Batch Operations & Holder Binding** - Issue multiple credentials simultaneously and cryptographically bind to holders vs single bearer token per flow
    - Bulk credential issuance for organizational use cases
    - Proof-of-possession verification to bind credentials to holder keys
    - Support for credential families and related credential sets
- **New/Modified Endpoints**:
    - `/.well-known/credential-issuer` - Credential issuer metadata discovery
    - `/credential` - The Credential Endpoint issues one or more Credentials of the same Credential Configuration and Credential Dataset upon presentation of a valid Access Token
    - `/deferred_credential` - This endpoint is used to issue one or more Credentials previously requested at the Credential Endpoint in cases where the Credential Issuer was not able to immediately issue this Credential
    - `/nonce` - This endpoint allows a Client to acquire a fresh c_nonce value
    - `/notification` - This endpoint is used by the Wallet to notify the Credential Issuer of certain events for issued Credentials
    - Enhanced `/authorization` and `/token` endpoints for credential-specific flows

8.2.3.2. Verifier Requirements

A Verifier is comparable to an OpenID Connect Relying Party (OIDC RP), but differs in several aspects:

- **Cryptographic Verification** - Must verify signatures vs trusting OP assertions
    - Validate VP signature instead of ID token signature
    - Direct signature validation using the issuer's published keys
    - Certificate chain validation and trust path verification
    - Complex credential combinations
- **Selective Disclosure** - Handle partial credential reveals vs full claim sets
    - Replace scope parameters with a Digital Credentials Query Language (DCQL) query
    - Processing of redacted or hidden claims in SD-JWT format
    - Zero-knowledge proof verification for privacy-preserving presentations
- **Presentation Requests** - Specify exactly what credentials are needed vs generic scopes
    - Support for alternative credential types and flexible acceptance criteria
    - Conditional logic for credential combinations and requirements
- **Offline Capability & Multiple Issuers** - Verify credentials without real-time OP contact
    - Cached issuer metadata and key material for offline verification

    - Trust registry integration for dynamic issuer acceptance
    - Deferred Responses
- **Nonce & Replay Protection** - Prevent presentation replay attacks vs session-based OIDC security
    - Generate and validate unique nonces for each presentation request
    - Timestamp validation to prevent stale credential presentations
    - Challenge-response mechanisms for fresh proof generation
- **New/Modified Endpoints**:
    - `/.well-known/verifier` - Verifier metadata discovery
    - `/request_uri` endpoint - Information about Verifier capabilities
    - `/response_uri` endpoint - Allow the Wallet to send the Authorization Response via an HTTP POST request
    - Enhanced `/authorization` endpoint supporting `response_type=vp_token`

8.2.3.3. Wallet Requirements

The Digital Identity Wallet acts as a user-controlled client, in contrast to server-side identity components. While the AARC Blueprint Architecture does not define wallet internals, the architectural patterns described in this document assume that the wallet provides the following capabilities:

- **User Sovereignty** - User controls their credentials vs OP-managed identity
    - User decides which credentials to accept and store
    - Complete control over credential sharing and presentation
    - No central authority can revoke, access, or modify user's credential collection
- **Credential Storage** - Persistent, encrypted credential vault
    - Long-term encrypted storage with backup and recovery mechanisms
    - Support for multiple credential formats and versions
    - Cross-device synchronisation while maintaining security
- **Selective Sharing** - User chooses what to disclose vs OP-determined claim release
    - Granular control over which claims to reveal in each presentation
    - Support for conditional disclosure based on verifier requirements
    - Privacy-preserving presentation with minimal data exposure
- **Key Management & Direct Interaction** - User owns cryptographic keys and can interact directly with other wallets vs OP-managed keys and mediated flows
    - Hardware-backed key generation and storage where available
    - Direct peer-to-peer credential exchange without intermediaries
    - Support for multiple key types and cryptographic protocols
- **New Endpoints**:
    - `/.well-known/wallet-config` - Wallet metadata discovery

- `/wallet/credentials` - Credential management API
- `/wallet/presentations` - Presentation handling endpoint
- `/wallet/attestations` - Wallet attestation services endpoint

8.3. Technical Trust Establishment and Metadata Handling

This section outlines how trust can be established when integrating wallet-based credential flows with AARC-compliant federated architectures.

8.3.1. Federated Trust Mechanisms

Trust in AARC-compliant federated architectures has grown to a complex, hierarchical, distributed, multi-national, fully meshed network of trust relationships. The SAML based identity federation eduGAIN is migrating towards the upcoming OpenID Federation [OID-Fed], because of scalability and complexity.

OID-Fed offers multiple mechanisms for establishing trust between two entities (e.g. an Issuer and a Verifier, an IdP and an SP, or an OP and an RP).

Trust Anchor

Trust Anchors are entities that are trusted, based on an external business decision. To evaluate whether a Proxy can be trusted, the trust chain is evaluated until it reaches a trusted root of the chain, the Trust Anchor. This structure allows modelling the hierarchical structure of today's eduGAIN.

Trust Marks

In addition, Trust Marks provide a separate structure for Trust Mark Issuers to make statements about entities. This separate structure allows modelling the different policies supported by federation members.

The [AARC-G100] guidelines for establishing trust between AARC-compliant AAI services using OpenID Federation (see Chapter 6), for example, considers two profiles. In a basic profile, two entities are supposed to trust each other, if they can establish a path to a Trust Anchor. In a more advanced profile, the OID-Fed feature of Trust Marks is used to assert specific policy support (e.g. [RAF], [SIRTFI], [DPCoCo]) to entities. This allows establishing a well informed trust decision in the complex domain of federated identity management.

8.3.2. Verifiable Credential-Based Trust Mechanisms

Digital Identity Wallets operate on a holder-centric trust model that differs from the operator-centric, server-to-server model of traditional federated AAIs. In the wallet ecosystem, the Verifier establishes trust directly with the Issuer of a Verifiable Credential rather than with an intermediate authentication provider.

Issuer Trust
Trust in an Issuer is typically anchored in pre-published trust sources rather than dynamic, bilateral metadata exchange. The main mechanisms are:

- the EU/EEA Trusted Lists [EU-Trust-List], which list accredited Qualified Trust Service Providers;
- decentralised issuer registries or DID resolution methods used in non-regulated deployments.

Cryptographic Verification
The Verifier validates the digital signature (or other cryptographic proof) on the Verifiable Credential using the Issuer's public key, retrieved from a trusted source (e.g. a trust list entry, a DID document, or an issuer metadata service such as an OpenID Federation entity configuration). This establishes authenticity and integrity independently of the transport protocol.

Credential Status
Credential validity beyond the cryptographic proof is determined via status mechanisms such as Bitstring Status List v1.0 [W3C-BSL] or issuer-specific revocation registries. These mechanisms are deliberately decoupled from the Issuer's immediate operational availability and, in many cases, support fully offline verification by the Verifier.

A key challenge of the current European digital identity (EUDI) for research collaborations is that the EU/EEA Trusted Lists [EU-Trust-List] are limited to nationally accredited providers and do not currently include the numerous Community AAIs/Collaboration Platforms or Infrastructure Proxies that legitimately issue research-specific credentials (membership, roles, entitlements, etc). This motivates the hybrid trust models described in Section 8.3.3.

8.3.3. Bridging the Two

To enable interoperability between AARC-compliant AAIs and the Wallet ecosystem, OpenID Federation 1.0 [OID-Fed] can serve as the common trust layer, as proposed in the OpenID Federation for Identity Wallet Architectures specification [OID-Fed-Wallet]. The development of the OpenID Federation Wallet Architectures 1.0 specification (Draft 03) was specifically guided by community feedback to prevent conflicts with existing OpenID wallet standards. The feedback led to the exclusion of any metadata parameter extensions in this draft that would affect other specifications, such as [OID4VP] and [OID4VCI], ensuring that [OID-Fed-Wallet] functions purely as a structural trust layer and not as a competing extension.

Approaches for bridging EU/EEA Trusted Lists with wallet-based architectures include direct consumption of Trusted List entries or using OpenID Federation as an intermediate trust layer. In practice, the latter approach does not require entities listed on EU/EEA Trusted Lists to directly implement OpenID Federation. A Trust Authority or federation operator may instead consume EU/EEA Trusted Lists and expose corresponding OpenID Federation metadata. In this model, EU/EEA

Trusted Lists remain the regulatory source of trust, while OpenID Federation provides the technical mechanism for trust discovery and metadata distribution.

In [OID-Fed-Wallet], the "Issuer" and "Verifier" roles of the Wallet ecosystem are treated as Federation Entities:

- **Discovering Trusted Issuers:** A Verifier (e.g. a downstream Proxy or service) can determine if a VC Issuer is legitimate by resolving the Issuer's **Entity Configuration**. If a valid Trust Chain exists from the Issuer to a recognised Trust Authority, the Issuer is trusted.
- **Metadata Distribution:** The OID-Fed Entity Configuration is used to distribute the specific metadata required for Wallet interactions, such as OID4VCI (Issuance) and OID4VP (Presentation) configurations, alongside standard OIDC metadata.
- **Policy Alignment:** The Fine-grained Trust Profile defined in [AARC-G100] leverages OID-Fed Trust Marks to express compliance and capabilities for AARC-compliant entities in a federation. Similar Trust Marks could be applied to VC Issuers. For example, a "Research Collaboration Issuer" Trust Mark could signal to a wallet or verifier that an entity is authorised to issue credentials for a specific scientific collaboration.

This approach allows Trust Authorities, as defined in [AARC-G100], to extend their scope to include Wallet Issuers and Verifiers without deploying separate infrastructure.

8.4. Open Technical Challenges

8.4.1. Schema Harmonisation

AARC deployments today rely on the AARC Attribute Profile [AARC-G056] and associated schemas such as eduPerson [EDUPERSON], SCHAC [SCHAC] and voPerson [VOPERSON]. VCs, however, permit more flexible claim structures, and without alignment the same attribute (e.g. affiliation or entitlements) may be encoded inconsistently across issuers. Ongoing work in the REFEDS VC Subcommittee is to explore mappings to represent R&E attributes in VC format [REFEDS-VC-WP]. Key challenges include:

- **Credential Representation:** Standardising how identifiers (persistent/pairwise, global), identity/authentication assurance, affiliation, and entitlements are represented in credential form.
- **Type Registration:** Establishing collision-resistant naming for credential types.
- **Identity Binding and Recovery:** A specific architectural challenge lies in the mapping between upstream "Core Identities" (such as the EUDI Wallet PID or an Institutional ID such as subject-id or eduPersonPrincipalName) and downstream collaboration identifiers. It is unclear whether a Collaboration VC must explicitly encode the upstream persistent identifier (e.g. voPersonID derived from EUDI) to ensure traceability.

- The Linking Problem: If the link between the eGov/Institutional ID and the Collaboration ID is maintained only via the user's wallet (i.e. "wallet binding"), the loss of the wallet (private key loss) breaks the chain of trust.
- The Recovery Problem: Guidelines are needed on whether collaboration VCs should embed the upstream personID to enable automated re-issuance and account recovery, or if privacy requirements dictate a strict separation that forces manual re-onboarding upon wallet loss.

8.4.2. Granular Disclosure and User Experience

Researchers often hold large sets of entitlements (collaboration membership, project roles, dataset access), creating both technical and User Experience (UX) challenges in Wallet-based flows [W3C-VC-UC].

- **Technical aspect:** Standard VC encoding may treat lists as indivisible. Mechanisms such as Selective Disclosure JWT (SD-JWT) allow claim-level disclosure, but applying selective disclosure to dozens or hundreds of entitlement values requires careful indexing, credential structuring, and caching strategies.
- **Usability aspect:** Presenting a large flat list of entitlements to the user leads to cognitive overload. Wallet UX should avoid situations where researchers must manually select from numerous VO entitlements.
- **Accessibility aspects:** Accessibility aspects include challenges related to the equal perception, understanding and management of verifiable credentials, their integration in the digital wallets, their functionality connected to selective disclosure and their privacy and security. For example, to avoid exclusion of persons with visual impairments, auditive alternatives for navigation/selection VCs should be included. Compatibility of assistive technologies already provided by the devices with the VCs might help to avoid accessibility inequalities. [W3C-WCAG3.0] .

Mechanisms such as Digital Credential Query Language (DCQL) in [OID4VP] could enable the Verifier to request only relevant entitlements, reducing disclosure and avoiding user decision fatigue.

8.4.3. Global Trust Interoperability

Research is inherently international, yet Digital Identity Wallet ecosystems are evolving along jurisdiction-specific trust frameworks. The EUDI Wallet under eIDAS 2.0 is a European model, but many countries operate or are developing their own government eID systems and digital wallets. A key challenge is enabling global participation in R&E collaborations, regardless of whether a user holds:

- an EUDI Wallet backed by the EU Trust List,
- a national government eID wallet outside Europe, or
- a research-issued (R&E) wallet using community trust roots.

To avoid fragmentation, research infrastructures need mechanisms that allow Verifiers to trust both government trust anchors and R&E trust frameworks.

8.4.4. Integration with Non-Web Workflows

Research workloads frequently operate outside a browser — HPC batch submission, CLI tools, SSH, Jupyter notebooks, delegation, API access. Current Wallet interactions assume a web-mediated UX, which is not always available in such environments. It is therefore essential to ensure that Wallet support is not limited to web-login use cases.

Pattern candidates include:

- OAuth 2.0 Device Flow adapted to Wallet-based attestation
- Token brokerage/robot credentials for long-running jobs
- VC-backed SSH certificates or OIDC/Oauth tokens minted post-verification

8.4.5. Token Translation

While Section 8.2 describes the mechanism where a Proxy verifies a VC and issues an Access Token, the open challenge lies in defining a standardised translation profile that ensures consistent behaviour across collaborations and infrastructures. Even if attribute schemas are harmonised (see Section 8.4.1), most services today consume OIDC tokens rather than VCs directly — and many are expected to continue doing so during the transition to wallet-native architectures. This raises a number of design questions that remain open for community alignment, including:

- How should token lifetime relate to VC expiry or revocation status?
- How should identity and authentication assurance from a VC propagate into AAI session context?
- How is replay prevented, and should Access Tokens be cryptographically bound to a Verifiable Presentation event?
- How should refresh or renewal occur without repeatedly invoking the Wallet?

9. Conclusions

The 2025 revision of the AARC Blueprint Architecture provides a reference architecture for implementing an AAI that supports common use cases within research collaborations.

AARC-BPA-2025 introduces a refined architectural description based on Functional Capabilities — Identity Management, Collaboration Management, and Service Integration. This capability-based view clarifies functional responsibilities while preserving the established component layer model. It supports modular deployment, clearer separation of concerns, and improved interoperability across independent BPA-compliant implementations. Importantly, its modular and capability-based structure lowers the entry barrier for small and medium research communities and collaborations, allowing them to adopt only the Functional Capabilities required for their use cases.

The architecture accommodates both the “Collaboration-first” approach (originally introduced in AARC-BPA-2019) and the new “User Identity-first with dynamic collaboration context” approach. The latter approach enables persistent identity establishment independent of collaboration context and supports dynamic retrieval of collaboration attributes when required. This approach can improve user experience in multi-collaboration environments and facilitates reuse of Identity Management services across infrastructures.

The separation of Identity Management from Collaboration Management also provides a foundation for future evolution towards credential issuance models, including verifiable credential representations. Such developments affect trust establishment and credential exchange mechanisms but remain compatible with the Functional Capability model defined in this document.

It should be stressed that AARC-BPA-2025 is backwards compatible with all previous versions of the BPA which have already been adopted by many e-infrastructure providers, research infrastructures and collaborations [AEGIS]. Existing AARC BPA deployments remain valid and interoperable, while the introduction of Functional Capabilities clarifies architectural structure without redefining the underlying deployment patterns.

The AARC Blueprint Architecture continues to provide a stable framework for federated access in research collaborations, supporting both current federation models and future trust and credential paradigms.

References

AARC-G027 AARC-G027 Guidelines for expressing resource capabilities, <https://aarc-community.org/guidelines/aarc-g027/>.

AARC-G045 AARC-G045 AARC Blueprint Architecture 2019, <https://aarc-community.org/guidelines/aarc-g045/>.

AARC-G052 AARC-G052 OAuth 2.0 Proxied Token Introspection, <https://aarc-community.org/guidelines/aarc-g052/>.

AARC-G056 AARC-G056 AARC Profile for Expressing Identity Attributes, Draft, <https://aarc-community.org/guidelines/aarc-g056/>.

AARC-G069 AARC-G069 Guidelines for expressing group membership and role information, <https://aarc-community.org/guidelines/aarc-g069/>.

AARC-G080 AARC-G080 AARC Blueprint Architecture 2025, <https://aarc-community.org/guidelines/aarc-g080/>.

AARC-G083 AARC-G083 Guidance for Notice Management by Proxies, <https://aarc-community.org/guidelines/aarc-g083/>.

AARC-G084 AARC-G084 Security Operational Baseline, <https://aarc-community.org/guidelines/aarc-g084/>.

AARC-G100 AARC-G100 Guidelines for Establishing Trust between AARC-compliant AAI services using OpenID Federation, Draft, <https://aarc-community.org/guidelines/aarc-g100/>.

AARC-I082 AARC-I082 Trust framework for proxies and Snctfi research services, <https://aarc-community.org/guidelines/aarc-i082/>.

AARC-I101 AARC-I101 Verifiable Credentials and Digital Identity Wallets in Research Collaborations, <https://aarc-community.org/guidelines/aarc-i101/>.

AEGIS AARC Engagement Group for Infrastructures, <https://aarc-community.org/about/aegis/>.

DPCoCo REFEDS Data Protection Code of Conduct (ver 2.0), <https://wiki.refeds.org/display/CODE/Data+Protection+Code+of+Conduct+Home>.

EDUPERSON eduPerson Object Class Specification, <https://wiki.refeds.org/display/STAN/eduPerson>.

EU-Trust-List EU/EEA Trusted List Browser: Trusted lists of qualified trust service providers in accordance with the EUDI Framework, <https://eidas.ec.europa.eu/efda/trust-services/browse/eidas/tls>.

EUDI-ARF European Commission, "European Digital Identity Wallet Architecture and Reference Framework", Version 2.7.3, November 2025, <https://eudi.dev/2.7.3/>.

OAuth-TT IANA, "OAuth Token Type Hints", <https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#token-type-hint>.

OID4VCI T. Lodderstedt, K. Yasuda, T. Looker, P. Bastian, "OpenID for Verifiable Credential Issuance 1.0", OpenID Foundation Final Specification, 16 September 2025, <https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html>.

OID4VP O. Terbu, T. Lodderstedt, K. Yasuda, D. Fett, and J. Heenan (eds.), "OpenID for Verifiable Presentations 1.0", OpenID Foundation Final Specification, 9 July 2025, <https://openid.net/specs/openid-4-verifiable-presentations-1_0.html>.

OIDC-Core OpenID Foundation. "OpenID Connect Core 1.0 Incorporating Errata Set 1". 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

OID-Fed Hedberg, R., Ed., Jones, M.B., Solberg, A.Å., Bradley, J., De Marco, G., and Dzhuvinov, V., "OpenID Federation 1.0", February 2026, <https://openid.net/specs/openid-federation-1_0.html>.

OID-Fed-Wallet De Marco, G., Hedberg, R., Jones, M.B., and J. Bradley, "OpenID Federation Wallet Architectures 1.0", OpenID Foundation Implementer's Draft 03, October 2024; <https://openid.net/specs/openid-federation-wallet-1_0.html>.

RAF REFEDS Assurance Framework, <https://refeds.org/assurance>.

REFEDS-VC-WP "White Paper on implementation of identifiers, attributes and claims for user identity in Research and Education", REFEDS VC Subcommittee Draft, December 2025, <https://docs.google.com/document/d/1b-Mlet3Lq7qKLEf1BnHJ4nL1fq-vMe7fzpXyrq2wp08/edit>.

RFC2119 Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

RFC6749 Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

RFC7159 Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

RFC7515 Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

RFC7518 Jones, M., "JSON Web Algorithms (JWA)", RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

RFC7519 Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

RFC7662 Richer, J., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/rfc/rfc7662>.

RFC8414 Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

RFC8693 Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/info/rfc8693>.

RFC8707 Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, <https://www.rfc-editor.org/info/rfc8707>.

RFC8725 Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

RFC9068 Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, <https://www.rfc-editor.org/info/rfc9068>.

RFC9864 Jones, M., "Fully-Specified Algorithms for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE)", RFC 9864, October 2025 <https://www.rfc-editor.org/info/rfc9864>.

RFC9901 Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JSON Web Tokens", RFC 9901, DOI 10.17487/RFC9901, November 2025, <https://www.rfc-editor.org/info/rfc9901>.

SCHAC SCHAC - SCHema for ACademia, <https://wiki.refeds.org/display/STAN/SCHAC+Releases>.

SHIB-NameIds Internet2 Shibboleth Project. "Name Identifier Formats", <https://wiki.shibboleth.net/confluence/display/CONCEPT/NameIdentifiers>.

SIRTFI REFEDS Security Incident Response Trust Framework for Federated Identity (SIRTFI), <https://refeds.org/sirtfi>.

VOPERSON voPerson v2.0.0, <https://github.com/voperson/voperson/blob/2.0.0/voPerson.md>.

W3C-BSL Sporny, M. and Herman, I. (eds.), "Bitstring Status List v1.0", W3C Recommendation, 15 May 2025, <https://www.w3.org/TR/vc-bitstring-status-list/>.

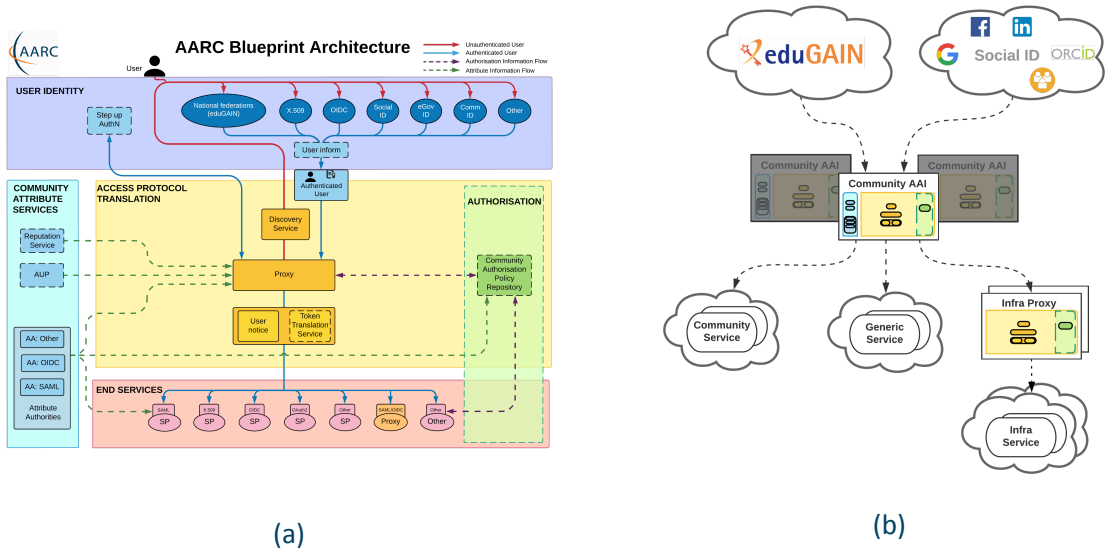
W3C-VC-UC W3C Working Group, Note, 24 September 2019 "Verifiable Credentials Use Cases", <https://www.w3.org/TR/vc-use-cases/#user-tasks>.

W3C-VC-DM2 Sporny, M., Longley, D., Sabadello, M., et al. (eds.), "Verifiable Credentials Data Model v2.0", W3C Recommendation, 15 May 2025, <https://www.w3.org/TR/vc-data-model-2.0/>.

W3C-WCAG3.0 Montgomery, R. B., Adams, C., Campbell, A., White, K., Spellman, J., Storr, F. (eds.), "W3C Accessibility Guidelines (WCAG) 3.0", <https://www.w3.org/TR/wcag-3.0/>.

Annex A: Revisions since AARC-BPA-2019

The current version of the AARC blueprint architecture builds upon the previous one AARC-BPA-2019 [AARC-G045] (depicted in Fig. A.1).



(a) (b)

Figure A.1: AARC Blueprint Architecture 2019 (AARC-BPA-2019)
(a) Component layers (b) AAI Service types

As described in Section 2.1, AARC-BPA-2025 retains five Component Layers.

The **User Identity Layer has been renamed to Authentication Layer** to more accurately reflect its specific role of housing authentication services. This avoids terminology overlap, as Identity Management is defined as a cross-cutting Functional Capability realised across multiple Component Layers — including Authentication, Access Protocol Translation, and Attribute Services — rather than being implemented within a single Layer. The **Authentication Layer also includes AARC BPA–compliant AAI proxies to enable chaining between AAI services.** The **Community Attribute Services Layer has been renamed to** simply **the Attribute Services Layer**, reflecting broader applicability beyond community-managed attributes. Similarly, the **Community Authorisation Policy Repository component has been renamed to Authorisation Policy Repository** to generalise its function. The **Reputation Service**, previously included in the architecture, **has been removed** in this revision due to limited adoption. The **User Inform component**, included in the Access Protocol Translation layer, **has been replaced by the Notice Presentation Component (NPC), and complemented by the introduction of a Notice & Policy Registry,** to support the notice presentation models defined in [AARC-G083].

Further updates in the current revision include the **introduction of the Identity Management Functional Capability** to group identity-related functions such as persistent identifier assignment, identity assurance, authentication assurance, and identity linking.

This capability acknowledges the increasing deployment of national identity services such as EDU-ID systems (The Netherlands, Switzerland, Germany, ...) that focus on the functionality of the Identity Management capability by offering persistent identity management and identity assurance capabilities. These systems, while often situated within national boundaries, are now considered key collaborators in a broader European identity ecosystem, including the upcoming EU Digital Identity Wallet paradigm (EUDI) and the European Open Science Cloud (EOSC) AAI.

In line with the capability model introduced in Section 2.2, the **current revision retains the functional scope of the Community AAI while mapping it to two of the newly defined Functional Capabilities: Collaboration Management and Identity Management.**

While its scope remains largely consistent in the Collaboration Management Functional Capability — managing user enrolment, roles, and collaboration-specific authorisation policies — the updated classification better reflects the diverse implementation contexts encountered by AAI operators. The previous term ("Community AAI") was often associated with formal, structured research communities, whereas the revised label accommodates a broader range of collaborative environments, including dynamic, project-based, or cross-institutional groups. Furthermore, this shift supports a **clearer architectural separation between identity management and collaboration management functionalities,** allowing implementers to build services aligned to capabilities that can evolve independently while remaining interoperable.

Lastly, the **general authorisation approach from AARC-BPA-2019 has been retained,** including group- and role-based entitlements [AARC-G069] and resource-specific entitlements [AARC-G027].

To enable token validation across infrastructures, a mechanism is required to allow an SP-IdP-Proxy acting as an Authorization Server (AS) to validate tokens issued by other trusted Authorization Servers. This is being formalised through the OAuth 2.0 Proxied Token Introspection specification [AARC-G052], which extends [RFC7662] by enabling an AS to proxy token introspection requests to the AS that originally issued the token, supporting validation of OAuth 2.0 tokens across trust domains.

Annex B: OpenID Federation Policies and Entity Examples

B.1. Federation Policies

This section provides examples of federation policies expressed through OpenID Federation Trust Marks in the context of [AARC-G100] (see Chapter 6).

B.1.1. Example Trust Mark Types and Issuers

Federation Policy	Trust Mark Type	Trust Mark Owner	Trust Mark Issuer
Sirtfi v1 & v2 [SIRTFI]	`https://refeds.org/sirtfi` `https://refeds.org/sirtfi2`	REFEDS / Sirtfi	National Identity Federation
REFEDS DP CoCo v2 [DPCoCo]	`https://refeds.org/category/code-of-conduct/v2`	REFEDS	self-issued
Security Operational Baseline [AARC-G084]	`https://aarc-community.org/guidelines/aarc-g084/`	AEGIS	self-issued
WISE Baseline AUP	`https://wise-community.org/wise-baseline-aup/v1/`	WISE SCI-WG	self-issued
REFEDS Assurance Framework v1 & v2 [RAF]	`https://refeds.org/assurance` `https://refeds.org/assurance/version/2`	REFEDS	self-issued
Snctfi	`https://aarc-community.org/snctfi`	AARC	self-issued

B.1.2. Metadata Policies

The following example illustrates a subordinate Entity Statement issued by a Trust Authority including metadata policy constraints in accordance with [OID-Fed].

```
{
  "exp": 1757495867,
```

```
  "iat": 1757409467,
  "iss": "https://ta.g100.2.example.com",
  "sub": "https://proxy.ri.example.org",
  "jwks": {
    "keys": [
      {
        "kid": "abc123",
        "kty": "RSA",
        "alg": "RS256",
        "use": "sig",
        "n": "....",
        "e": "AQAB"
      }
    ]
  },
  "metadata_policy": {
    "federation_entity": {
      "contacts": {
        "essential": true
      },
      "display_name": {
        "essential": true
      },
      "organization_name": {
        "essential": true
      },
      "policy_uri": {
        "essential": true
      }
    },
    "openid_provider": {
      "client_registration_types_supported": {
        "essential": true,
        "superset_of": [
          "automatic"
        ]
      },
      "id_token_signing_alg_values_supported": {
        "subset_of": [
          "RS256",
          "ES256",
          "ES512"
        ]
      },
      "scope": {
        "essential": true,
        "superset_of": [
          "openid",
          "profile",
          "email"
        ]
      },
      "subject_types_supported": {
        "essential": true,
        "superset_of": [
```

```
          "public"
        ]
      },
      "userinfo_signing_alg_values_supported": {
        "subset_of": [
          "RS256",
          "ES256",
          "ES512"
        ]
      }
    }
  }
}
```

B.2. Entity Configuration Examples

This section provides decoded examples of Entity Configurations for selected Entity Types as defined in [OID-Fed].

B.2.1. Trust Authority

```
{
  "exp": 1757495867,
  "iat": 1757409467,
  "iss": "https://ta.g100.2.example.com",
  "jwks": {
    "keys": [
      {
        "alg": "ES256",
        "crv": "P-256",
        "kid": "xFYuOlyKBlMghUMke-x-fvKm17Pub3LowiWe-szNsa8",
        "kty": "EC",
        "use": "sig",
        "x": "SqpIPSH6n7RcY4ZWco8_i5XMeoGh8LrWoTmxfASEfFM",
        "y": "rPiFQrTLPfaLk1zG4zxUgVm-ZptAZEYBp6Rfoc2H6Rc"
      }
    ]
  },
  "metadata": {
    "federation_entity": {
      "display_name": "AARC G100.2 Example TA",
      "federation_fetch_endpoint": "https://ta.g100.2.example.com/fetch",
      "federation_list_endpoint": "https://ta.g100.2.example.com/list",
      "federation_resolve_endpoint": "https://ta.g100.2.example.com/resolve",
      "organization_name": "Example Organisation"
    }
  },
  "sub": "https://ta.g100.2.example.com",
  "trust_mark_issuers": {
    "https://refeds.org/sirtfi": [
      "https://tmi.example.com"
    ],
    "https://refeds.org/sirtfi2": [
      "https://tmi.example.com"
```

```
    ]
  }
}
```

B.2.2. Trust Mark Issuer

```
{
  "authority_hints": [
    "https://ta.g100.2.example.com"
  ],
  "exp": 1757574701,
  "iat": 1757488301,
  "iss": "https://tmi.example.com",
  "jwks": {
    "keys": [
      {
        "alg": "ES256",
        "crv": "P-256",
        "kid": "njbnIq34F9BLD2jAGnZV4J0-IPwFu7dEq0CAEM05qOo",
        "kty": "EC",
        "use": "sig",
        "x": "dx4wzrVuWN9GpSKhIzz1kXq5YzFfFTL5syp3Y4IR9CU",
        "y": "rGVM5yYgg8PnSXauuX1VACXzOWDDXx0OKJBHf4m7BOs"
      }
    ]
  },
  "metadata": {
    "federation_entity": {
      "display_name": "Trust Mark Issuer",
      "federation_trust_mark_endpoint": "https://tmi.example.com/trustmark",
                                              "federation_trust_mark_list_endpoint":
"https://tmi.example.com/trustmark/list",
                                            "federation_trust_mark_status_endpoint":
"https://tmi.example.com/trustmark/status",
      "organization_name": "Example Organization"
    }
  },
  "sub": "https://tmi.example.com"
}
```

B.2.3. Proxy with OP and RP roles

```
{
  "iss": "https://proxy.ri.example.org",
  "sub": "https://proxy.ri.example.org",
  "iat": 1756375124,
  "exp": 1756461524,
  "jwks": {
    "keys": [
      {
        "kid": "abc123",
        "kty": "RSA",
        "alg": "RS256",
        "use": "sig",
        "n": "....",
        "e": "AQAB"
      }
```

```
    ]
  },
  "metadata": {
    "openid_provider": {
      "issuer": "https://proxy.ri.example.org",
      "authorization_endpoint": "https://proxy.ri.example.org/oidc/auth",
      "token_endpoint": "https://proxy.ri.example.org/oidc/token",
      "userinfo_endpoint": "https://proxy.ri.example.org/oidc/userinfo",
      "jwks_uri": "https://proxy.ri.example.org/oidc/jwks",
      "response_types_supported": [
        "code",
        "id_token",
        "code id_token"
      ],
      "subject_types_supported": ["public", "pairwise"],
      "id_token_signing_alg_values_supported": ["RS256"],
      "scopes_supported": [
        "openid",
        "profile",
        "email",
        "offline_access",
        "entitlements",
        "voperson_external_affiliation",
        "schac_home_organization"
      ],
      "claims_supported": [
        "sub",
        "name",
        "given_name",
        "family_name",
        "preferred_username",
        "email",
        "email_verified",
        "acr",
        "eduperson_assurance",
        "entitlements",
        "voperson_id",
        "voperson_external_affiliation",
        "schac_home_organization"
      ],
      "client_registration_types_supported": [
        "automatic",
        "explicit"
      ],
      "federation_registration_endpoint": "https://proxy.ri.example.org/oidfed/reg"
    },
    "openid_relying_party": {
      "application_type": "web",
      "redirect_uris": [
        "https://proxy.ri.example.org/callback",
        "https://proxy.ri.example.org/callback2"
      ],
      "client_name": "Example RI",
      "client_uri": "https://proxy.ri.example.org",
      "jwks_uri": "https://proxy.ri.example.org/oidc/jwks",
```

```
      "response_types": ["code"],
      "grant_types": [
        "authorization_code",
        "refresh_token"
      ],
      "token_endpoint_auth_method": "client_secret_post",
      "id_token_signed_response_alg": "RS256",
      "scope": "openid profile email entitlements",
      "client_registration_types": [
        "explicit"
      ],
      "tos_uri": "https://proxy.ri.example.org/aup"
    },
    "federation_entity": {
      "contacts": ["aai-support@ri.example.org"],
      "organization_name": "Example RI",
      "logo_uri": "https://proxy.ri.example.org/assets/logo.png",
      "policy_uri": "https://proxy.ri.example.org/privacy"
    }
  },
  "authority_hints": [
    "https://interm1.example.org",
    "https://ta1.example.net"
  ]
}
```