17-12-2025

Deliverable D4.1:

Validation Results report

Publication Date	17 December 2025
Due Date:	Month 22 (December 2025)
Authors:	Allan Norré Pedersen (NORDUnet), Anders Sjöström (NORDUnet/SUNET), Chryssa Thermolia (GRNET), David Groep (Nikhef NWO-I), Diana Gurdu (KIT), Konstantinos Georgilakis (GRNET), Marcus Hardt (KIT), Martin van Es (SURF), Nicolas Liampotis (GRNET), Patrick Machalet (CERN), Peter Lényi (MU), Peter Thijsse (MARIS) and Themis Zamani (GRNET)
Contributors:	Fernando Gonzalez Perez (EGI), Björn Erik Abt (PSI)
Version:	1.0
Document Code:	AARC TREE D4.1
Publishing Organisation:	NORDUnet/SUNET
DOI:	NA

Abstract
This report presents the results of the pilots and validation activities conducted within Work Package 4 (Adoption and Validation) of the AARC TREE project. It describes the testing of interoperability guidelines, the development and deployment of an automated validator suite, and feedback from participating Research Infrastructures (RIs). The report provides insights into technical feasibility, adoption challenges, and recommendations for future improvements.

Table of Contents

1. Introduction

The AARC-TREE project extends the Authentication and Authorisation for Research and Collaboration (AARC) programme by delivering updated guidance, architectural patterns, and operational recommendations that enhance interoperability across Research Infrastructures (RIs). As RI ecosystems adopt increasingly federated, cross-infrastructure identity and access control models, consistent implementation of the AARC guidelines is essential.

Work package 4 “Adoption and Validation” worked closely with WP1 (Technical Guidance and Architecture) and WP2 (Trust Policy Harmonisation and Interoperability) to validate the guidelines developed within AARC TREE. This was achieved by designing and executing cross Research Infrastructure (RI) pilots to assess how the guidelines can be applied in a real environment and confirm their technical feasibility.
In addition, the work package delivered an automated validator suite and collaborated with AEGIS, research and e-infrastructure forum to maintain and evolve AARC results, for the implementation of an online validation service.

This deliverable presents the findings of the three major guideline validation pilots and reports on validator suite and associated tooling. It provides concrete implementation feedback based on the pilots’ activities within RI’s and the development and deployment of the validator suite and the usage thereof.

2. Piloting Activities

Within Task 1 in WP4, the piloting activities focus on validating key aspects of the updated AARC-TREE guidelines by applying them to concrete use-cases from Research Infrastructures. The G056 pilot on the AARC Attribute Profile examines how stable, non-reassignable identifiers and other identity attributes can be managed and exchanged across infrastructures to support consistent and secure user recognition. The G083 notice management pilot explores how RIs can handle user information notices—such as consent, attribute release notifications, or operational messages—in a standardised and interoperable way. Complementing these, the G100 OpenID Federation pilot tests approaches for establishing trust between AAI services using OpenID Federation-based mechanisms, ensuring that infrastructures can rely on a flexible and scalable model for cross-domain authentication and authorisation.

2.1 Pilot on AARC Attribute Profile (AARC-G056)

The pilot was originally scoped as “subject identifiers pilot” aimed at validating the implementation of AARC-G056 for handling Public and Pairwise Subject Identifiers across AARC-compliant authentication and authorisation environments. During execution, the scope broadened to validate the full attribute profile, including identifiers, name, email, affiliation, assurance information,

group/role entitlements, and resource capabilities. Pairwise identifiers were removed from the AARC-G056 profile during this process, shifting the focus to public subject identifiers and to the wider attribute profile. The pilot ensured that public subject identifiers conform to AARC-G056 requirements, including uniqueness, scope handling, representation across protocols, and compatibility with relying-party software.

2.1.1 Objectives

The pilot validated compliance with AARC-G056 focusing on:

- Conformance with the draft AARC-G056 attribute profile across SAML and OIDC representations.
- Public subject identifier propagation through multi-proxy environments.
- Relying party (RP) compatibility across common libraries.
- Conformance to mandatory identifier release rules.

2.1.2 Methodology

- The pilot included multiple proxy implementations operating in different roles, including Identity Proxies, Community AAIs, and Infrastructure Proxies. Participating test relying parties included:
- `mod_auth_openidc` (OIDC RP)
- `Keycloak` (as SAML SP and OIDC RP)

The test environment also included a Mujina SAML Identity Provider functioning as a Home IdP, used to perform controlled attribute release tests.

Validation scope:

- SAML 2.0 Attribute Statements
- OIDC ID Token, UserInfo, Access Token
- Attribute release across IdP → Proxy → RP paths
- Multi-proxy/chain flows (where supported)
- Public subject identifiers (sub/subject-id and voperson_id/voPersonID)
- Pairwise subject identifiers were initially considered, but removed from the AARC-G056 profile during the pilot

2.1.3 Results

- Subject identifiers (`subject-id` and `voPersonID` / `voperson_id`) were validated across all implementations.
- Migration to `sub` as the sole public subject identifier was not technically feasible for all proxies, due to software limitations in some proxies and the requirement to preserve stable identifiers for connected relying parties. In particular, `sub` values

were previously only unique per issuer and changing their value to global identifiers would break backward compatibility.

- Based on this outcome, WP1 updated the draft AARC-G056 definition of the public subject identifier to allow either `sub` or `voperson_id`, pending further validation.
- For SAML flows, scope validation behaviour was confirmed: proxies -depending on their role- may need to validate the scope against the issuer's metadata; RPs must not be required to do so.
- Both Keycloak and mod_auth_openidc were able to consume `voperson_id` as the subject identifier using appropriate configuration, and can likewise be configured to use `sub` where preferred. Example configurations for both RP implementations are provided in Annex A.
- Per-proxy outcome for subject identifiers
    - EGI Check-in releases scoped sub and voperson_id
    - Helmholtz ID releases scoped voperson_id and unscoped sub
    - LS AAI releases scoped sub and voperson_id
    - GEANT Core AAI releases scoped sub and voperson_id
    - Indigo IAM releases sub and voperson_id unscoped
- In the proxies participating in the pilot, `voperson_id` usually requires a separate `voperson_id` scope, although the guideline mandates that it must always be released, no matter what scope is requested

The following table checks if the OIDC claims specified in AARC-G056 are released correctly for the proxies included in this pilot. This is based on the mod_auth_openidc RP, and verifies claims in Access Token, ID Token, and from the userinfo endpoint. Limitation: the RP is not able to check what is released at the introspection endpoint.
The checkmarks show where the claims are detected for each proxy, and the green background marks whether a claim is present in all the sources required by the guideline and thus whether the proxy complies with the guideline for that specific claim.
SeaDataNet has adopted the EGI Check-In OpenID connect login method to replace an older AAI system. With this adoption comes the usage of AARC attributes in practice, for example in the user profile.

\ Proxy	RCIAM Keycloak (EGI Check-In, e.g. in SeaDataNet)			Unity IdM (Helmholtz ID)			Indigo IAM			GEANT Core AAI			Perun AAI (LS AAI)		
Attribute \ Where	**AT**	**ID**	**UI**	**AT**	**ID**	**UI**	**AT**	**ID**	**UI**	**AT**	**ID**	**UI**	**AT**	**ID**	**UI**
sub	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
voperson_id		✔	✔			✔	✔	✔	✔	✔		✔		✔	✔

name			✓			✓			✓			✓		✓	✓
given_name			✓			✓			✓			✓		✓	✓
family_name			✓			✓			✓			✓		✓	✓
email			✓			✓			✓	✓		✓		✓	✓
organization_name															
schac_home_organization			✓												✓
voperson_external_affiliation			✓			✓			✓	✓	✓	✓		✓	✓
eduperson_assurance	✓		✓			✓	✓	✓	✓						✓
entitlements			✓			✓	✓	✓	✓	✓		✓			✓
aarc_ver															
voperson_policy_agreement															

2.1.4 Gaps Identified

- Inconsistent release of `voperson_id`.
- Missing validation of provenance and case-sensitivity rules.
- Missing validation of `aarc_ver` profile version attribute.

2.1.5 Recommendations

- Proxy implementations should ensure release of `voperson_id` when required by relying parties, rather than only issuing it when the `voperson_id` scope is explicitly requested.
- Add SCIM and LDAP mapping tests.
- Validate case-sensitivity and provenance metadata.
- Proxy implementations should ensure continued alignment with the AARC Architecture working group as AARC-G056 matures, particularly as attributes like `aarc_ver` and `voperson_policy_agreement` are finalised.

2.2 Pilot on Notice Management (AARC-G083)

This pilot aimed to validate the AARC-G083 guidelines for notice management within AARC-compliant Authentication and Authorisation Infrastructure (AAI) environments. It focused on the implementation, presentation, and logging of user information notices (such as Acceptable Use Policies, privacy policies, and other notices) in AARC-compliant environments, particularly using machine-readable notices and aggregated presentation components within proxies. The aim was to reduce the number of clicks and interstitial screens, facilitating non-interactive workflows.

2.2.1 Objectives

The pilot evaluated:

- Aggregation of machine-readable notices (WISE AUP(s), privacy policies, service terms).
- Interoperable presentation through a Notice Presentation Component (NPC).
- Logging and assertion of user acceptance.
- Registration, management, and versioning of notices.
- Interoperability with SAML and OIDC authentication.
- Support for multi-proxy and non-interactive flows.

2.2.2 Methodology

The pilot deployed:

- A Notice Registry for structured storage of service notices.
- An NPC capable of aggregating notices and presenting them to users.
- Service-provider components integrated with federated login (SAML and OIDC).
- Logging services recording acceptance events.
- JSON-based machine-readable notice formats aligned with WISE Baseline AUP.
- The registry is hosted by CERN: https://base-api-notice-management-api.app.cern.ch/[1]
- The NPC is hosted by SURF: https://npc.pt-76.utr.surfcloud.nl/ [2]

2.2.3 Results

- Notice aggregation worked consistently across services.
- The NPC displayed merged notices, respecting order and versioning rules.
- Acceptance logging was consistent and interoperable.
- Policies could be updated at the registry without impacting SP implementations.
- Machine-readable notices followed a stable schema and validated correctly.

[1] See annex B for examples of registry endpoints
[2] See annex B for examples of use

- Multi-proxy flows required additional harmonisation of redirection practices.
- Offline access presented challenges; additional non-interactive notice flows are needed.

2.2.4 Gaps Identified

- No unified JSON schema catalogue for common notice types.
- Limited validation of user experience consistency across NPC deployments.
- Missing normative error-handling behaviour at the registry level.
- Legal interoperability (GDPR compliance) requires clearer metadata fields.
- Limited testing with external community AAIs.

2.2.5 Recommendations

- Publish canonical JSON schemas for all notice types.
- Define UX guidance for NPC implementations.
- Standardise error codes and failure-handling.
- Add optional legal metadata fields for GDPR context.
- Expand notice-management testing to more RIs.

2.3 Pilot on Guidelines for Establishing Trust between AARC-compliant AAI services using OpenID Federation (AARC-G100)

This pilot aimed to validate the AARC-G100 Guidelines for Establishing Trust between AARC-compliant AAI services using OpenID Federation. It focused on testing the implementation and configuration of SP-IdP-Proxies, in line with the OpenID Federation specification, in order to establish trust without direct bilateral trust relationships.

2.3.1 Objectives

The goal of the pilot was to validate both trust profiles defined in AARC-G100:

- Basic Trust Model (G100.1): trust established solely through membership in the federation by constructing trust chains to a recognised Trust Authority.
- Fine-Grained Trust Model (G100.2): trust established through federation membership and additional constraints expressed via Trust Marks and metadata policies.

For both profiles, the pilot assessed:

- Publication of OIDC federation metadata (entity configurations) by participating proxies.

- Construction and validation of trust chains via the Trust Authority.
- Explicit and automatic client registration workflows for proxies acting as OpenID Providers (OPs).
- Explicit client registration workflows for proxies acting as Relying Parties (RPs).
- Application of Trust Marks and metadata policies in trust resolution (G100.2 trust profile).

2.3.2 Methodology

The pilot setup included:

- Two Trust Authorities, one for each trust profile (G100.1 and G100.2), acting as Trust Anchor and Immediate Superior.
- Four proxy implementations publishing federation-compliant metadata and registered with the relevant Trust Authority: GÉANT Core AAI Platform, Indigo-IAM, Perun AAI (LS AAI), and RCIAM Keycloak (EGI Check-in — e.g. SeaDataNet)
- Test OPs and RPs supporting automatic and explicit client registration.
- A Trust Mark Issuer for issuing Trust Marks used in G100.2 (e.g., Sirtfi v1/v2).

Proxies executed authentication flows in both OP and RP roles and performed trust establishment across proxies as well as the test OPs and RPs.

2.3.3 Results

At the time of the pilot, OpenID Federation support across available AAI software stacks was still limited: four proxy implementations were able to participate, with others planning support but not yet ready for validation. Further deployment details and the federation topology can be found in Annex C. The main results are summarised as follows:

- Proxies successfully published entity configurations and registered with the Trust Authorities
- Trust chain construction and validation for G100.1 worked as expected across participating implementations.
- Explicit client registration was validated across all participating implementations.
- Automatic client registration was validated for most implementations in their OP role, while automatic registration in the RP role is not yet available.
- OP and RP roles were validated based on the Authorisation Code flow between proxies.
- Support for G100.2—Trust Marks and metadata-policy evaluation—remains limited across all proxies, and none fully implemented all required features.

2.3.4 Gaps Identified

- Limited software support for the fine-grained trust model (G100.2): none of the participating proxy implementations fully support Trust Mark validation or metadata-policy enforcement as required by the G100.2 profile.
- Automatic client registration is not supported by any of the participating proxy implementations in their RP role.
- Ecosystem maturity: only a small set of software stacks currently support OpenID Federation, which constrains broader interoperability testing.
- Need for clearer guidance on how explicit client registration should be triggered and managed in OIDC Federation deployments, including the role of OP discovery mechanisms.

2.3.5 Recommendations

- Prioritise implementation of Trust Mark validation and metadata-policy support required for G100.2.
- Extend support for automatic client registration to include the RP role.
- Enhance robustness of metadata update handling in RP implementations.
- Continue interoperability testing to validate both OP and RP roles across trust profiles.

3. Validation Tools

WP4 collaborated with WP1 and WP2 to develop automated validation tools. A broad range of validation tools already used across the research and education community was reviewed. A complete inventory is provided in [AARC-TREE-M4.1]. While these tools address specific aspects such as metadata validation or attribute conformity, none offer a validation framework capable of assessing both technical behaviour and the operational and policy requirements defined in the AARC guidelines.

To address this, WP4 defined requirements for a new Validator Suite (see Annex D). The suite was designed to support automated and self-assessment-based validation, operate as an online service, allow extension to new guidelines, and ensure that assessment results remain private unless explicitly shared.

The resulting Validator Suite developed in WP4 is based on the Compliance Assessment Toolkit (CAT), which enables guideline requirements to be expressed as principles, criteria, automated tests, and structured self-assessment questions.

A dedicated CAT instance for AARC TREE is deployed by GRNET at: https://aarc3.cat.argo.grnet.gr/

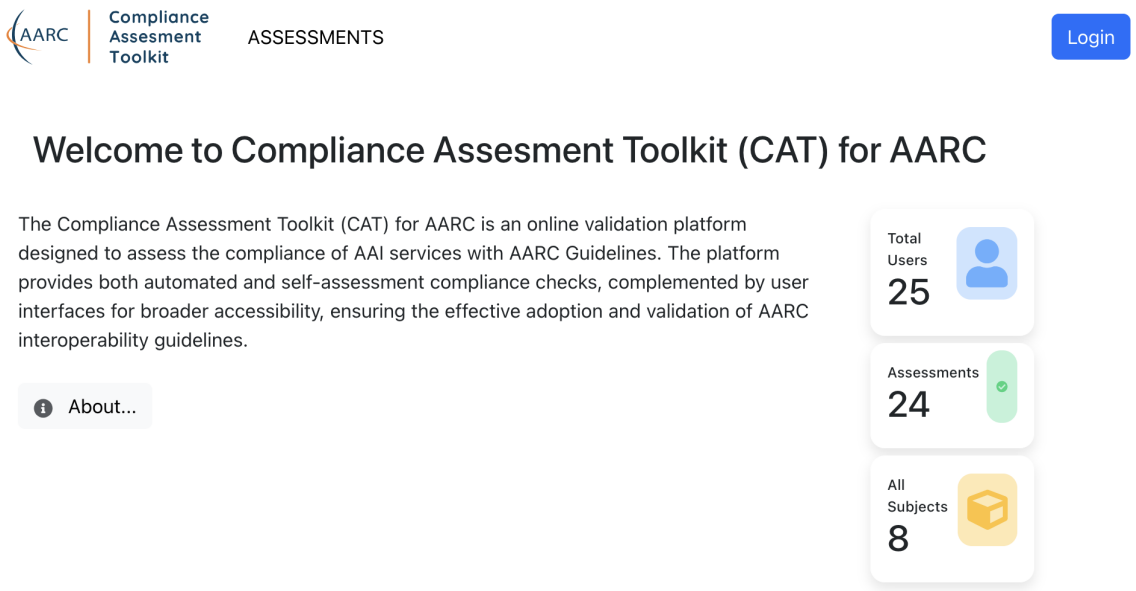

Figure 1— Compliance Assessment Toolkit (CAT) for AARC

CAT is a specialised tool/framework designed to help organisations assess whether their services, infrastructure, or practices comply with a set of standards, norms, or policies. It was originally built for operational assessment of compliance with the policy of the European Open Science Cloud (EOSC) — especially around persistent identifiers (PID) policy compliance — but the design is deliberately reusable in many contexts. CAT aims to support broader quality-assurance needs such as alignment with community expectations like FAIR principles, TRUST principles, CARE principles, interoperability frameworks, participation requirements in federated systems, and other formal specifications. Within WP4, CAT was extended accordingly, and three AARC guidelines were modelled and validated in the instance for AARC TREE::

- AARC-G069 (Specification for Expressing Group and Role Information) — Demonstrates automated validation of claim presence and formatting (e.g. URN syntax of group entitlements according to AARC-G069).
- AARC-G056 (AARC Profile for Expressing Identity Attributes) — Uses checks to validate multiple attributes such as subject identifiers, affiliations, assurance information, and entitlements.
- AARC-G071 (Secure Operation of Attribute Authorities) — Implemented as a structured self-assessment for non-automatable operational requirements.

These examples illustrate CAT's capacity to support both automated attribute-level checks and operator-assessed organisational requirements. Example assessment outputs for AARC-G056 and AARC-G069, and the self-assessment questions for AARC-G071, are provided in Annex E.

Many AARC guidelines require validation of behaviour across multiple OpenID Connect (OIDC) endpoints (e.g. UserInfo and Token Introspection) and tokens. To support this, WP4 extended naco, the NFDI Attribute COnformity Checker, to expose an API that allows CAT to retrieve and inspect OIDC protocol artefacts for applying AARC guideline-specific rules (e.g. presence of mandatory claims in endpoints or tokens and claim value syntax)

This integration provides a scalable workflow for onboarding new AAI services. The process consists of:

- registering the AAI service in naco, and
- adding the service in CAT and assigning a service-provider organisation.

3.1 User Workflow

Users logging into CAT are recognised as *Identified Users* and may only view public assessments. To gain permissions within the service, users must submit a Validation Request, selecting their role and AAI Service provider from the list offered by naco. Because this request requires administrative approval, an authorised administrator reviews the submission and decides whether to accept or reject it; in both cases, the user receives an email notification with the decision. A *Validated User* is someone formally recognised within the system and authorised to contribute to external evaluations, assessments, and potentially self-assessments.

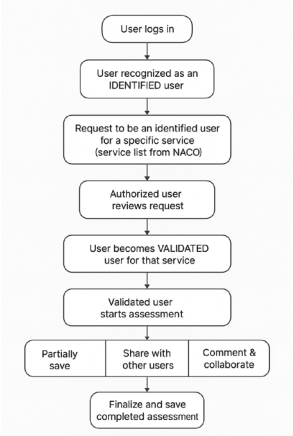

Figure 2 — Compliance Assessment Toolkit (CAT) User Workflow

Validated Users can choose between manual and automated assessments, save their progress incrementally, share assessments with other users, and collaborate through the integrated commenting system. After completing all required sections, they can finalise the assessment and obtain a report summarising the results.

3.2 Features

Some of the main features of CAT are the following:

General Features

- Support for multiple guidelines (e.g. AARC-G069, AARC-G056, AARC-G071)
- Support for repeated assessments of the same AAI Service Provider against the same guideline
- Manual and automated test execution
- Detailed assessment reports
- Search and filtering with multiple criteria
- Import, download and export of assessments in an exchange format

Collaboration Features

- Commenting on assessments
- Sharing assessments with other validated users
- Adding evidence and descriptive justification to answers
- Usage statistics

Administrative Features

- Dashboard overview
- Reporting tools
- Search, filter and view assessments
- Search, filter and view validation requests
- System statistics
- Assessment management

The following sections provide details on several core features of the service.

3.2.1 Assessment Builder

The CAT Assessment Builder is the core component used to design, customise, and manage compliance assessments in a structured way. Through the Assessment Builder, users can define questions, scoring, and evidence requirements, while linking each item to specific principles, guidelines, or policy obligations. Overall, it functions as a practical bridge between high-level compliance expectations and their measurable, operational implementation.

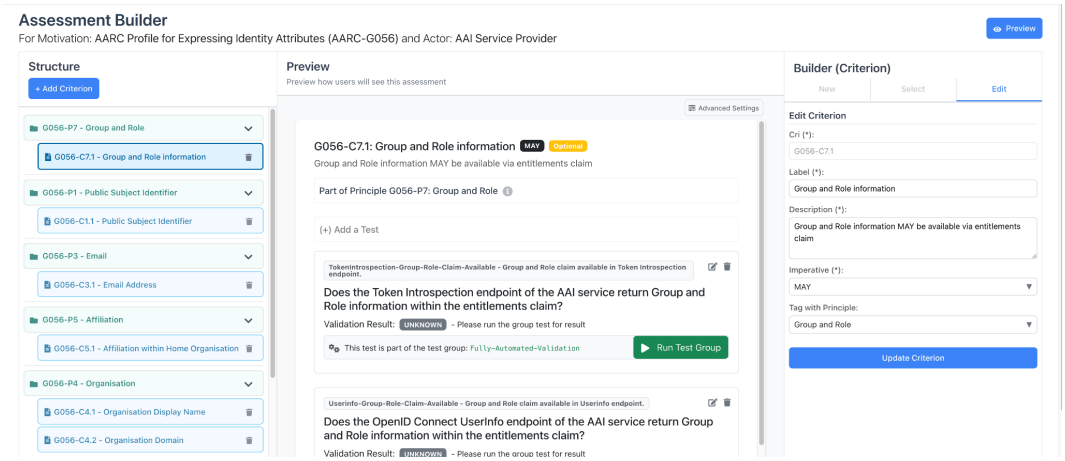

Figure 3 — Compliance Assessment Toolkit (CAT) Assessment Builder

3.2.2 Report Builder

The Report Builder enables authorised users to view the results of the assessments submitted by service owners.

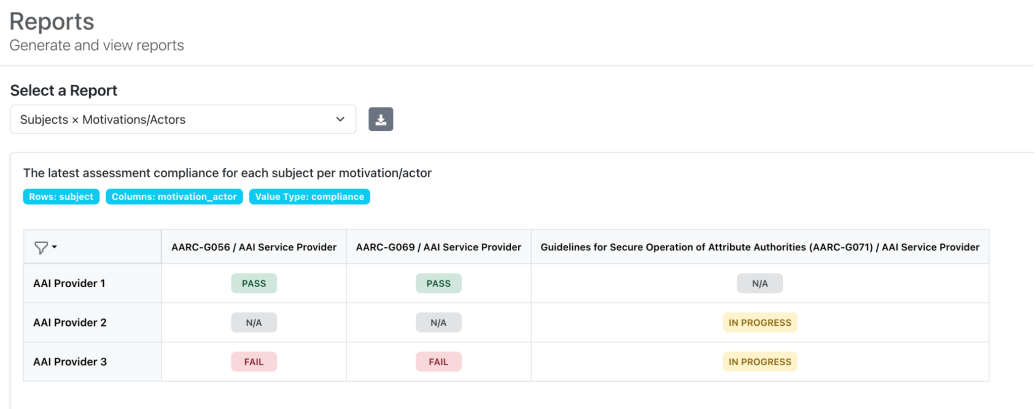

Figure 4— Compliance Assessment Toolkit (CAT) Report Builder

With the Report Builder, users with the appropriate permissions can:

- Browse all assessments completed by service owners
- Export or share those reports for governance, audits, or follow-up actions

In short, the Report Builder turns individual compliance assessments into actionable insights for oversight and coordination.

3.2.3 Dashboard

The Dashboard provides a centralised overview that supports administrators in monitoring and managing the system. It is primarily designed for observation, giving a high-level view of user activity, role assignments, validation workflows, and assessment statistics. While operational actions such as editing, validating, or deleting data are performed through the dedicated management sections, the Dashboard allows administrators to quickly identify trends and issues. It also includes a view of the distribution of answers—especially failed or non-compliant ones—helping highlight areas that may require further attention.

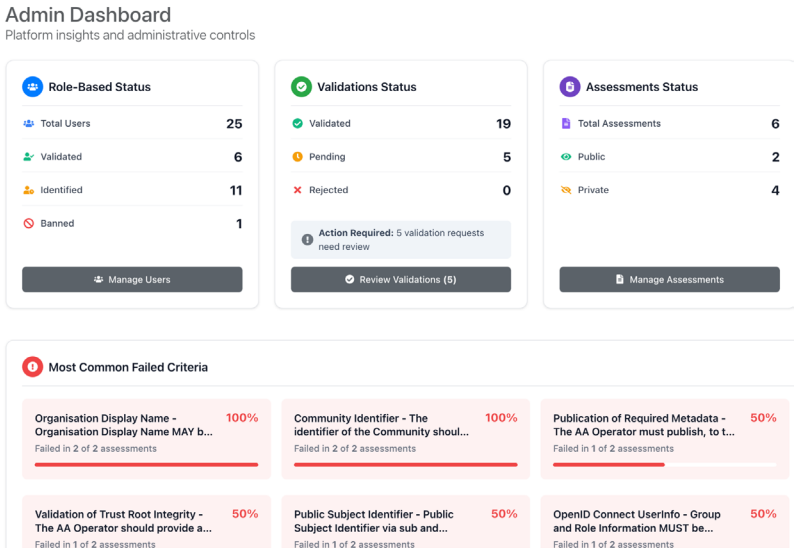

Figure 5— Compliance Assessment Toolkit (CAT) Admin Dashboard

3.3 Feedback

For guidelines that validate identity attributes, such as AARC-G056 and AARC-G069, participants noted that automated checks depend on the attributes available for the specific user evaluated through naco. If the selected user lacks certain attributes (e.g. group entitlements), these will not appear in the retrieved protocol artefacts and will therefore be reported as absent in CAT. This highlights the importance of using representative test accounts during validation.

Participants who completed the AARC-G071 self-assessment provided additional feedback on the usability of the questionnaire. The assessment required approximately 1.5 hours to complete, and in several cases required consulting the guideline text to resolve ambiguities in the interpretation of specific questions. Considering the complexity of the guideline and the number of operational requirements it covers, this duration was not unexpected, although it highlights the value of continued efforts to streamline guidance and supporting materials for future assessments. The use of structured identifiers (e.g. “AN-1”) was viewed positively, as it facilitated cross-referencing with the guideline document. While internal comments are already supported, participants suggested that a per-question comment field would better support documenting how responses were determined.

4. Results and Analysis

The three pilots and the validation tools collectively show that:

- AARC-G056 is implementable with high interoperability for identifiers.
- AARC-G083 is viable but requires additional schema standardisation.
- OID-Fed adoption is promising but ecosystem maturity varies.

Cross-cutting trends:

- Proxies behave consistently across both SAML and OIDC.
- Relying-party client libraries differ significantly, documentation is essential.
- Interoperability increases where guidelines are explicit and schema-based.
- OpenID Federation metadata policies and Trust Marks can unify security expectations across infrastructures.
- Automated validator checks help surface attribute-release inconsistencies early, especially for AARC-G056 and AARC-G069.
- Structured identifiers in guidelines (e.g. AARC-G0710) improve traceability between guideline requirements and validation results.

Remaining weaknesses:

- Lack of consistent error-handling across protocols.
- Limited adoption of SCIM/LDAP for user provisioning and deprovisioning.
- Evolving attribute definitions (e.g. new attributes in AARC-G056 such as `aarc_ver`) and changes to subject identifiers pose challenges for consistent implementation and for maintaining backwards compatibility.
- Multi-proxy flows are high-value but need more formalisation.
- Support for OpenID Federation Trust Marks and metadata-policy enforcement (Fine-grained Trust Model G100.2) is not yet available across implementations.

5. Conclusions and Recommendations

The pilots successfully showed that the guidelines are sound and workable. The tools developed within the workpackage are currently hosted by the partners (CERN, GRNET, KIT and SURF) for the sustainability that need to be formalised through an agreement to extend the operation beyond the project lifetime. In the meantime, the CAT-based Validator Suite and its integration with naco, is available and can continue to be used by RIs and AAI service operators for guideline validation and self-assessment. WP4 demonstrated that such tooling can be operated as a shared service; however,

modelling additional AARC guidelines and further development of the tooling would require sustained community effort and appropriate funding.

The piloting and validation of the AARC guidelines illustrated that:

- Tooling is extended, this remains central to adoption.
- Schema and metadata definitions continue to be strengthened.
- Deployment guidance is expanded for RIs and community AAIs.
- Vendor ecosystems are encouraged to support AARC semantics.

Key recommendations:

- Maintain and extend validation tooling beyond the project lifetime.
- Strengthen schema-based sections of guidelines (G056, G083).
- Publish authoritative interoperability checklists.
- Provide training and deployment templates for RIs.
- Continue cross-infrastructure pilots to validate new features.

With the evolution of the Open ID federation and other AARC guidelines there will be a need for further validation of these frameworks. This workpackage has shown that the validation of the guidelines within the AARC community is a necessary exercise for validation and pointing out weak spots in the guidelines. A future AARC project focussing on the currently developed frameworks (OpenID federations and wallets) would be ideally suited to verify the alignment of the implementations with the frameworks.

Annexes

Annex A: AARC-G056 AARC Attribute Profile pilot

Configuration Guide for Relying Parties

OIDC Relying Parties

Keycloak

If you are using Keycloak as an OpenID Connect Relying Party, then you need to follow the steps below in order to configure an AARC-G056 compliant Infrastructure Proxy as an Identity Provider based on OpenID Connect:

1. Access the administrator console of your Keycloak instance and navigate to **Identity Providers** and then select **OpenID Connect v1.0**.

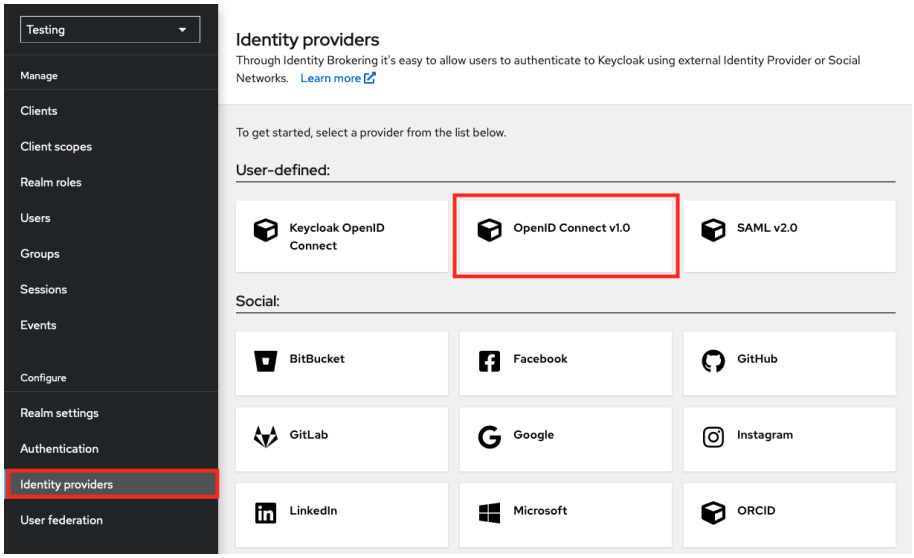

Figure 6 — Keycloak configuration for connecting to an AARC-G056-compliant OpenID Connect Identity Provider

2. After configuring the client credentials for the Identity Provider, scroll down to the **OpenID Connect settings** section and expand the **Advanced** option and then add the required **Scopes**. For example:

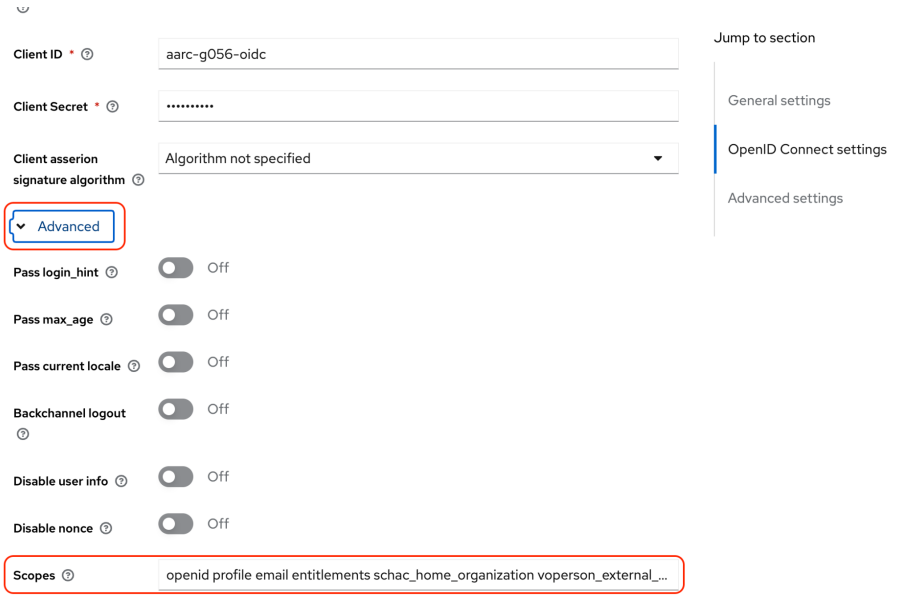

Figure 7 — Keycloak configuration for requesting OpenID Connect scopes following the AARC-G056 profile

3. Next, you will need to add the mappers to store the required claims. Go to the **Mappers** tab and then click on **Add Mapper**.

Figure 8 — Keycloak configuration for mapping user attributes to OpenID Connect claims following the AARC-G056 profile

To map the Subject Identifier through the sub claim, you will need to add the following options:

```
Name: username
Sync Mode Override: import
Mapper Type: Username Template Importer
Template: ${CLAIM.sub}
Target: LOCAL
```

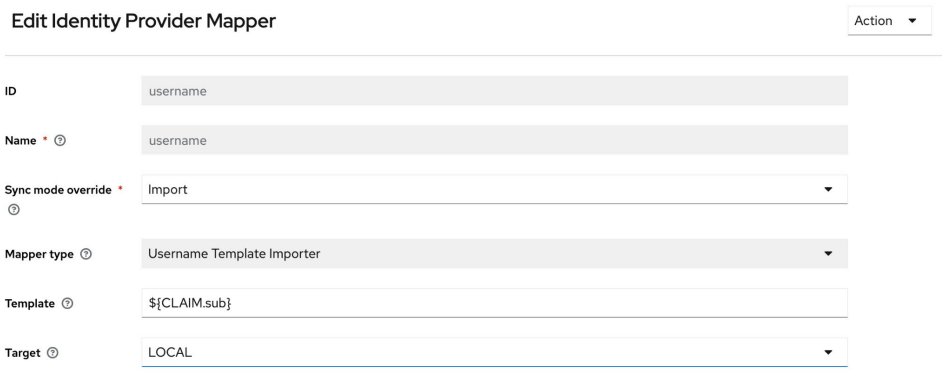

Figure 9 — Keycloak configuration for mapping the AARC-G056 Public Subject Identifier through the OpenID Connect sub claim

Alternatively, you can map the Subject Identifier through the voperson_id claim by adjusting the configuration as follows:

```
Name: username
Sync Mode Override: import
Mapper Type: Username Template Importer
Template: ${CLAIM.voperson_id}
Target: LOCAL
```

mod_auth_openidc

If you are using the mod_auth_openidc module for the Apache web server to operate as an OIDC relying party, then you need to follow the steps below in order to configure an AARC-G056 compliant proxy as an Identity Provider.

1. Modify your Apache virtual host to add the OIDC client credentials and scopes that the service needs. The example below assumes that EGI Check-in (dev instance) is the Identity Provider, and that the restricted path that requires authentication is under /secure.

```
<VirtualHost *:443>
        ...
        OIDCProviderMetadataURL https://aai-dev.egi.eu/auth/realms/egi/.well-
known/openid-configuration
        OIDCClientID <client id obtained when registering at OP>
        OIDCClientSecret <client secret obtained when registering at OP>
        OIDCProviderTokenEndpointAuth client_secret_basic
        OIDCRedirectURI <the redirect URI you registered at the OP for the client>
        OIDCCryptoPassphrase <generate a strong password>
        OIDCScope "openid email profile aarc entitlements eduperson_entitlement
voperson_id"

        <Location /secure>
```

```
            AuthType openid-connect
            Require valid-user
        </Location>
</VirtualHost>
```

2. Next, you need to select the **sub** claim for identifying the user, which will set the value for the REMOTE_USER variable or header.

```
        OIDCRemoteUserClaim sub
```

3. Alternatively, you can use the **voperson_id** claim in the same way:

```
        OIDCRemoteUserClaim voperson_id
```

4. You can pass all claims received from the OP, either as environment variables, or HTTP headers, or both (shown below). You can also set a prefix for the names of these variables or headers.

```
OIDCClaimPrefix "OIDC_CLAIM_"
OIDCPassClaimsAs both
```

5. The variables or headers are then available for use in your web application. For example, in PHP you can retrieve them in the following way:

```
        echo $_SERVER['OIDC_CLAIM_sub']
        $hdrs = apache_request_headers();
        echo $hdrs['OIDC_CLAIM_sub'];
```

`mod_auth_openidc` also supports multiple OIDC Identity Providers, accessible via a discovery page. This is done by configuring the directory where all provider metadata files are located, via OIDCMetadataDir. You need to create three files for each provider:

- `<urlencoded-issuer-url-with-protocol-and-trailing-slashes-stripped>.provider`: this file contains metadata retrieved from the well-known endpoint of the OP
- `<urlencoded-issuer-url-with-protocol-and-trailing-slashes-stripped>.client`: contains client configuration, such as credentials and response type
- `<urlencoded-issuer-url-with-protocol-and-trailing-slashes-stripped>.conf`: contains client specific mod_auth_openidc configurations, such as scopes

SAML Service Providers

Keycloak

If you are using Keycloak as a SAML Service Provider, then you need to follow the steps below in order to configure an AARC-G056 compliant Infrastructure Proxy as a SAML Identity Provider:

1. Access the administrator console of your Keycloak instance and navigate to **Identity Providers** and then select **SAML v2.0**.

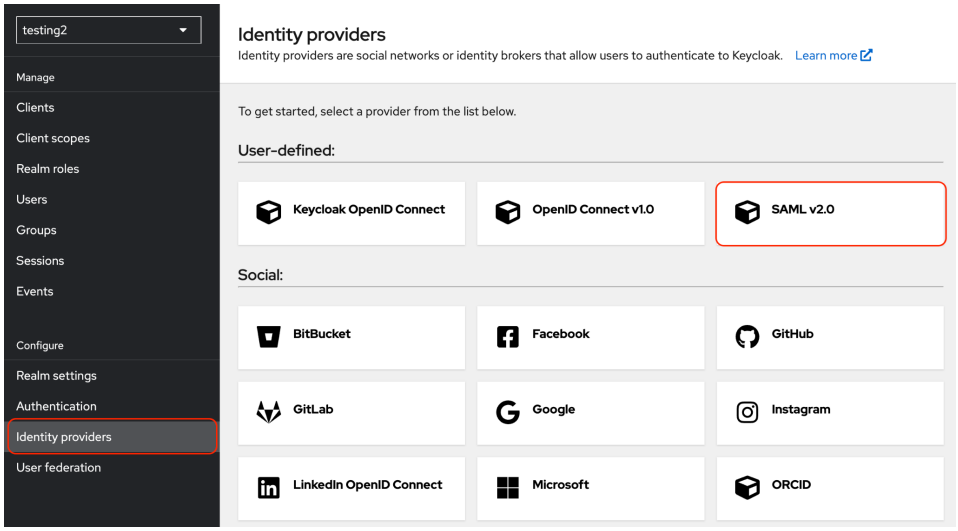

Figure 10 — Keycloak configuration for connecting to an AARC-G056-compliant SAML v2.0 Identity Provider

2. Configure the SAML IdP by setting an alias, using your realm's SAML SP entity ID, and importing the IdP metadata via the entity descriptor

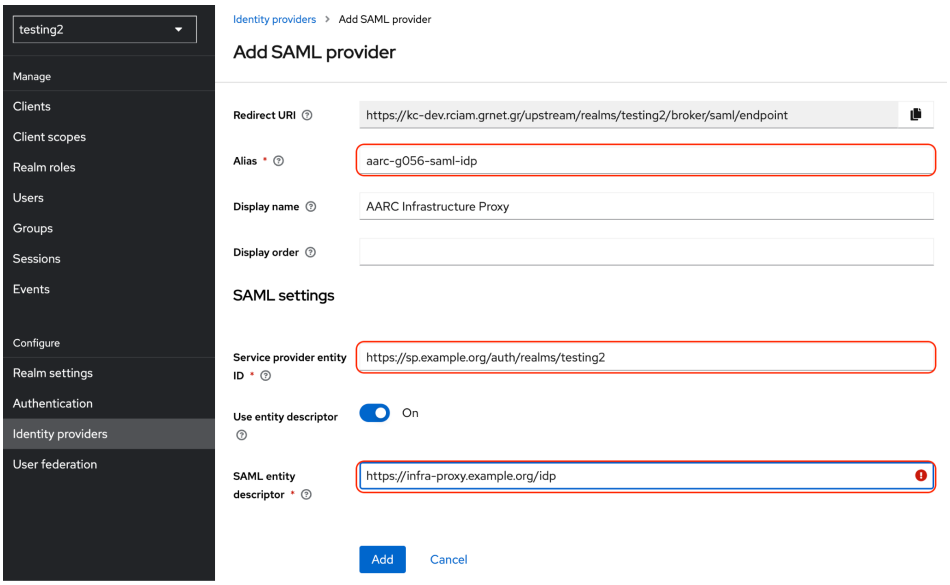

Figure 11 — Keycloak configuration for connecting to an AARC-G056-compliant SAML v2.0 Identity Provider using metadata

3. After importing the SAML IdP metadata, navigate to the **SAML settings** of the IdP configuration and configure the **Principal type** (select Attribute [Name]) and **Principal attribute** (set to urn:oasis:names:tc:SAML:attribute:subject-id). This will map the received subject-id SAML attribute value to the Username property of the Keycloak user:

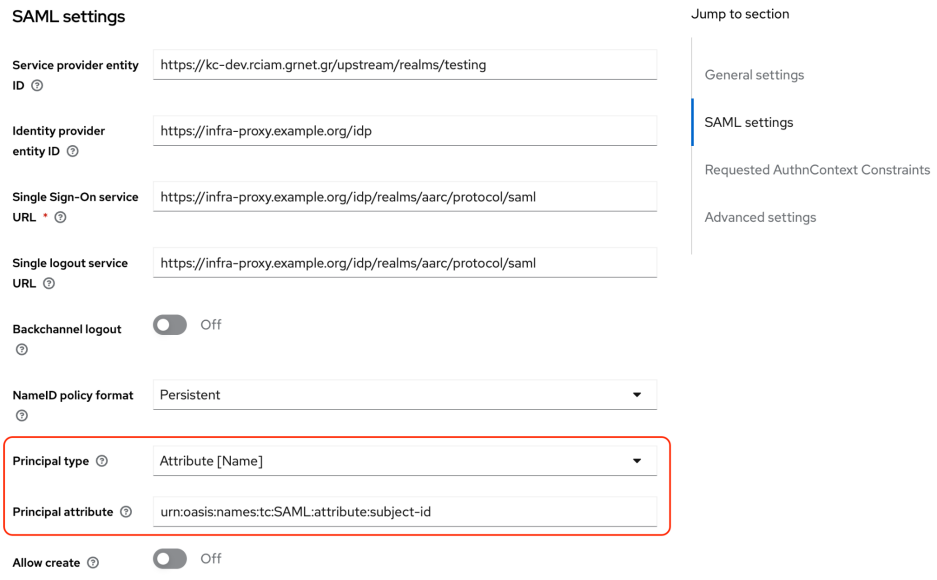

Figure 12 — Keycloak configuration for mapping the AARC-G056 Public Subject Identifier through the subject-id SAML attribute

4. Next, you will need to add the mappers to store the required SAML attributes. Go to the **Mappers** tab and then click on **Add Mapper**.
5.

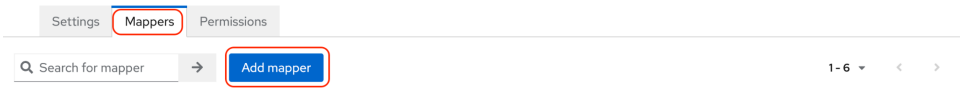

Figure 13 — Keycloak configuration for mapping user attributes to SAML attributes following the AARC-G056 profile

To map the Subject Identifier through the voPersonID SAML attribute, configure the following options:

```
Name: voPersonID
Sync Mode Override: Import
Mapper Type: Attribute Importer
Attribute Name: urn:oid:1.3.6.1.4.1.25178.4.1.6
Friendly Name: voPersonID
Name Format: ATTRIBUTE_FORMAT_URI
User Attribute Name: voPersonID
```

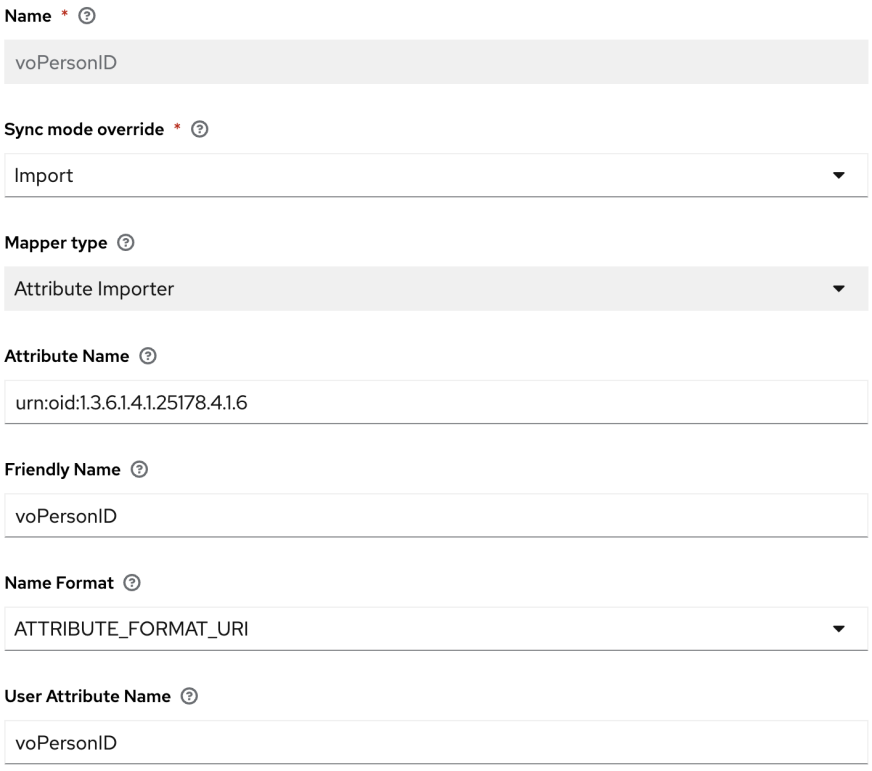

Figure 14 — Keycloak configuration for mapping the AARC-G056 Public Subject Identifier through the voPersonID SAML attribute

Repeat the process to create Attribute Importer mappers for all required SAML attributes, (e.g. givenName, sn, email, eduPersonEntitlement, voPersonExternalAffiliation, schacHomeOrganization).

Annex B: G083 components and implementations

Steps and Components

The WISE AUP model uses a layered approach to AUP composition:

- the Baseline AUP is a set of ten 'commandments' *(see appendix)* that are identical and equally applicable to all services and infrastructures,
- the body or bodies that are authoritative to grant access , as well as the purpose binding of the user's activities are *templated* explicitly in the preamble,
- the AUP provides a place for optional additional agreements or terms and conditions, that *augment* (but not replace or contradict) the Baseline AUP commandments. For communities 'hosted within' one or more infrastructures, such additional agreements can be combined (stacked) and presented once, and together.

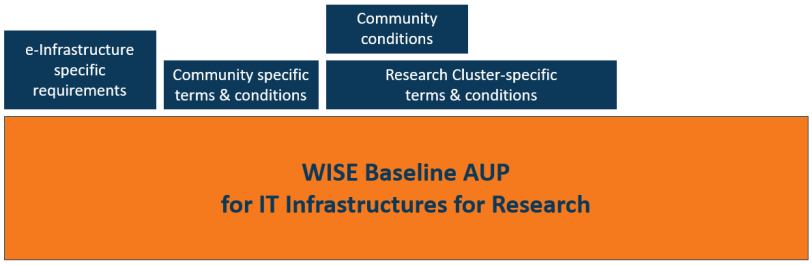

Figure 15 — The baseline Acceptable Use Policy (AUP) hierarchy

The notices within a community concern the upper (blue) part shown in diagram 1.

WISE AUP (template)

Acceptable Use Policy and Conditions of Use

This Acceptable Use Policy and Conditions of Use ("AUP") defines the rules and conditions that govern your access to and use (including transmission, processing, and storage of data) of the resources and services ("Services") as granted by {community, agency, or infrastructure name} for the purpose of {describe the stated goals and policies governing the intended use}.

<To further define and limit what constitutes acceptable use, the community, agency, or infrastructure may optionally add additional information, rules or conditions, or references thereto, here or at the placeholder below. These additions must not conflict with the clauses 1-10 below, whose wording and numbering must not be changed.>

1. You shall only use the Services in a manner consistent with the policies and for the purposes described above, show consideration towards other users, and collaborate in the resolution of issues arising from your use of the Services.
2. You shall only use the Services for lawful purposes and not breach, attempt to breach, nor circumvent administrative or security controls.
3. You shall respect intellectual property and confidentiality agreements.
4. You shall protect your access credentials (e.g. passwords, private keys or multi-factor tokens); no intentional sharing is permitted.
5. You shall keep your registered information correct and up to date.
6. You shall promptly report known or suspected security breaches, credential compromise, or misuse to the security contact stated below; and report any compromised credentials to the relevant issuing authorities.
7. Reliance on the Services shall only be to the extent specified by any applicable service level agreements listed below. Use without such agreements is at your own risk.
8. Your personal data will be processed in accordance with the privacy statements referenced below.
9. Your use of the Services may be restricted or suspended, for administrative, operational, or security reasons, without prior notice and without compensation.
10. If you violate these rules, you may be liable for the consequences, which may include a report being made to your home organisation or to law enforcement.

<Insert additional numbered clauses here.>

The administrative contact for this AUP is: {email address for the Granting Authority}
The security contact for this AUP is: {email address for the infrastructure, community, and/or Granting Authority security contact}
The privacy statements (e.g. Privacy Notices) are located at: {URL}
Applicable service level agreements are located at: {URLs}

Blue text are mandatory to change
Green text is optional to change
Black text must be kept

For the presentation of the notices the following steps have been identified:

1. **Machine-Readable Notice Aggregation**
    - Implement support in proxies to collect notices from connected services.
    - Present notices as a single, combined screen during user authentication.

- Policy identifiers are fetched from a registry and versioned; acceptance is time stamped and asserted downstream via the *voPersonPolicyAgreement* or *voperson_policy_agreement claim (multivalued).*

2. **Configuration of the Notice Presentation Component (NPC)**
   - The NPC acts as the frontend for user notice presentation.
   - JSON-based metadata defines each notice[3].
   - If the same notice version was previously accepted, it is not shown again.
     A. **Scenario: OIDC with offline_access**
        - Proxy or service indicates the need for persistent access (refresh tokens).
        - This is shown in the initial notice bundle alongside other conditions.
     B. **Scenario: GDPR Notice Presentation**
        - Policies are categorised as "privacy" or "conditions".
        - The presentation point must identify the data controller for each notice.
        - User acceptance is logged with timestamp and version information.
3. **Fallback and Notice Updates**
   - When notices are updated or at the *notice_refresh_period* (if present), they are re-presented at the next user session.
   - Alternatively, users are notified via email where possible.

Community and entities relations

There are three different relationships between the community and the entity responsible for the notice presentation component (NPC). These are illustrated in Figures 16—18:

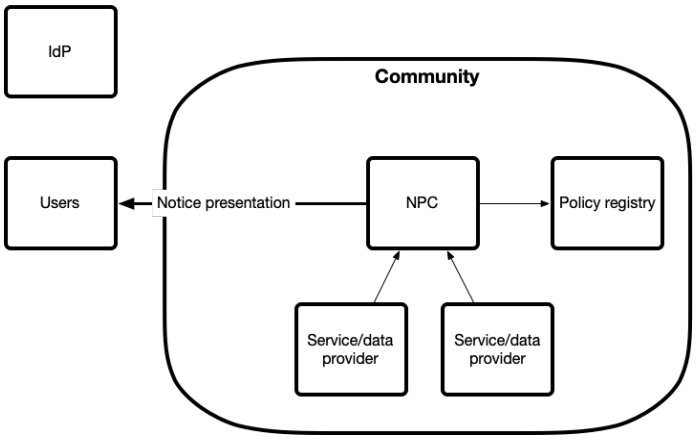

Figure 16 — Community and NPC entity are by construction the same entity

[3] See: *Guidance for notice management by proxies* section 5 (DOI: 10.5281/zenodo.14452340)

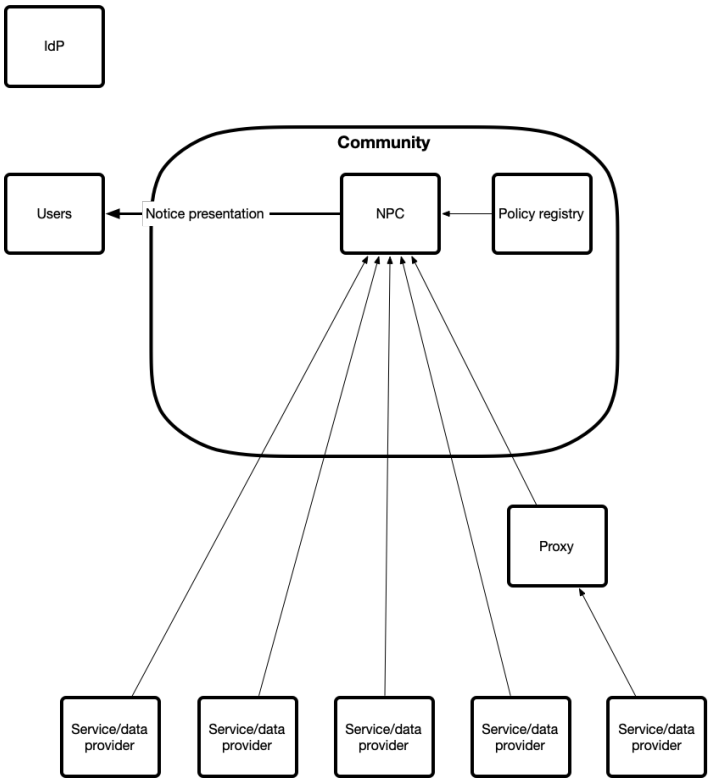

Figure 17 — The NPC is run on behalf of and under the control of the community

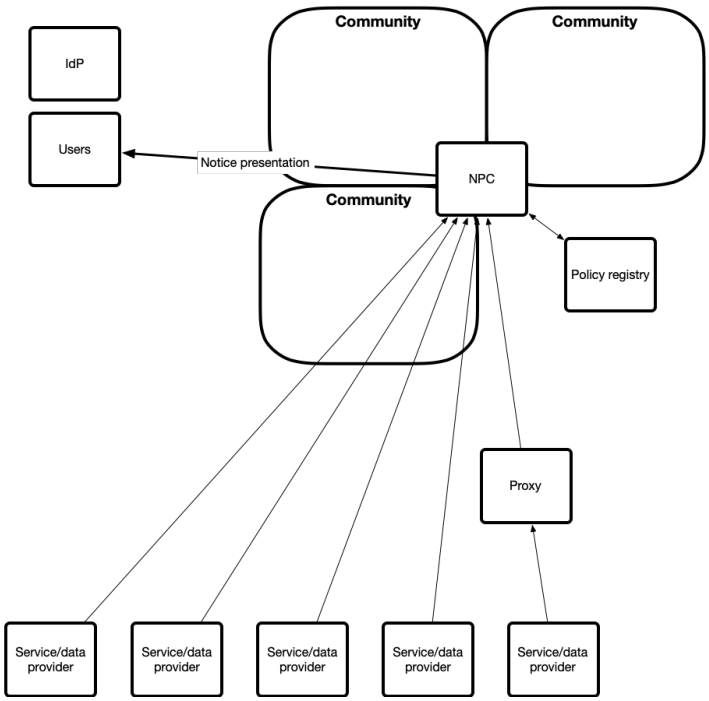

Figure 18 — The NPC collects information itself, and offers to host communities on its platform

Policy registry implementation

1. Schema for policy identifier entries:
    - **URI**
    - **name** for human presentation
    - **informational URL** pointing to a JSON document stored at the registry as well
    - **owner** some way to designate the service or person that added the policy
2. JSON doc schema:
    - **id**: URI string
    - **aut**: URI identifiying authority governing policy
    - **aut_name**: plain text name of authority
    - **valid_from**: time from which policy is in effect (changed on major and minor changes)
    - **ttl**: time period after which json doc should be retrieved again by consumer
    - **contacts**
    - **security_contacts**
    - **privacy_contacts**
    - **policy_class**: string from the limitative enumeration ('purpose', 'acceptable-use', 'conditions', 'sla', 'privacy'); qualify with jurisdiction using #, e.g.: privacy#eu
    - **notice_refresh_period**: seconds after which this notice has to be presented again to the user, regardless of earlier acceptance
    - **includes_policy_uris**: URIs of policies that are implicitly fulfilled
    - **augments_policy_uris**: URIs of policies that are augmented by this policy, e.g. WISE AUP -> NPC may need to merge them to this policy when presenting to user
    - **policy_url**: URL of policies in human readable form
    - **description**: short plain text human readable description -> used for presentation in composite notices
3. registry API
    - public
        - `get_policies`
            - returns json array of json objects containing: `uri, name, informational_url`
    - protected with API key / AT for pre-approved services that can add policies to the registry
        - `add_policy`
            - input: uri, name, json doc
            - actions
                - checks that json doc has valid schema
                - generate informational url where json doc will be stored
                - add uri to id in json document

        - store policy entry
    - returns
        - success: uri, name, informational_url
        - error
- get_api_key[4] ?
- register_service[5] ?

Similar to registry for assurance profiles: https://www.iana.org/assignments/loa-profiles/loa-profiles.xhtml

Implementation

Figure 19 illustrates the interactions between the User, AAI Proxy, Notice Presentation Component, and Policy Registry during the authentication and notice presentation flow.

[4] *Were not implemented in the pilot as the focus was on tokens in api keys*
[5] Were not implemented in the pilot as the register service could not be fully automated.

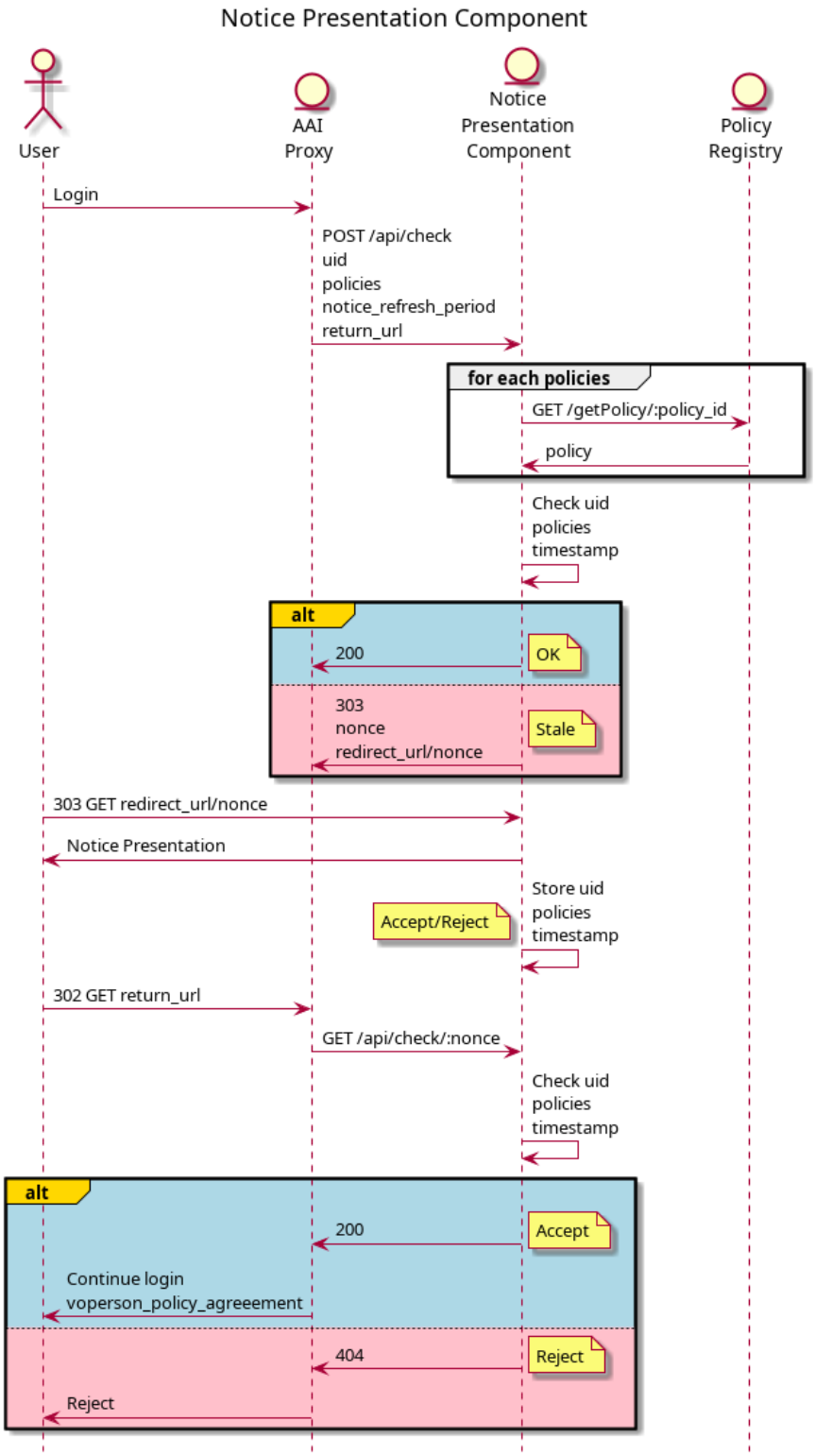

Figure 19 — Notice Presentation Component flow diagram

Example of the Notice Presentation for some dummy policies:

In this example, policy *abcd* is included from *1234* and *ijkl* is included from abcd. *efgh* is also included from *1234*. *ijkl* includes *1234* and causes a loop but that is detected and not shown.

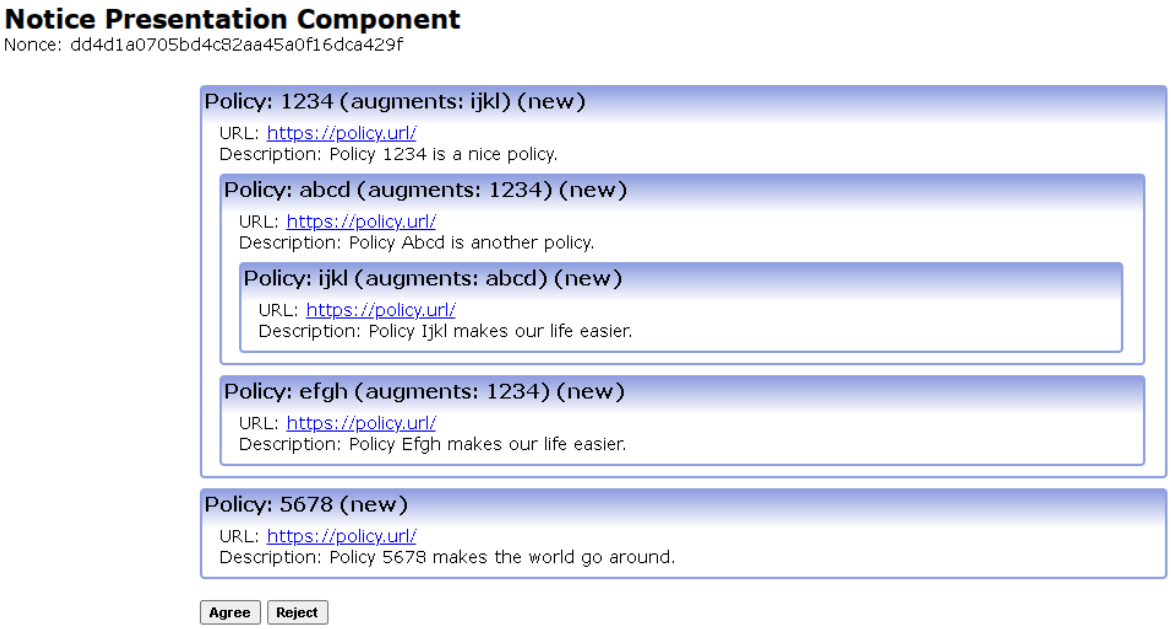

Figure 20 — Example of Notice Presentation

The NPC can be tested here: https://npc.pt-76.utr.surfcloud.nl/

Notice Presentation Component implementation

```
curl --request POST \
  --url https://npc.pt-76.utr.surfcloud.nl/api/check \
  --header 'Content-Type: application/json' \
  --data '{
      "uid": "test",
      "policies": [
            "https://another-community.org", "https://global-
science.org/policy, urn:idk:123456"
      ],
      "notice_refresh_period": 60,
      "return_url": "http://localhost:8081/return"
}'
```

Result:

```
{
  "nonce": "645ad5d10549475a918201e2a0420aac",
  "redirect_url": "https://npc.pt-
76.utr.surfcloud.nl/npc/645ad5d10549475a918201e2a0420aac"
}
```

Visit the above redirect_url (different for each request)

Then, after agreeing:

```
curl --request GET \
  --url https://npc.pt-
76.utr.surfcloud.nl/api/check/645ad5d10549475a918201e2a0420aac
```

And inspect the result:

```
{
  "voperson_policy_agreement": [
    "https://another-community.org",
    "https://some-community.org",
    "urn:doi:10.60953/68611c23-ccc7-4199-96fe-74a7e6021815",
    "urn:idk:123456"
  ]
}
```

Registry implementation

```
curl --request GET \
  --url https://base-api-notice-management-api.app.cern.ch/getPolicies
```

Result (abbreviated):

```
{
  "policies": [
    {
      "id": "https://another-community.org",
      "informational_url": "https://base-api-notice-management-
api.app.cern.ch/getPolicy/https%3A%2F%2Fanother-community.org",
      "name": "Another community"
    },
    {
      "id": "https://global-science.org/policy",
      "informational_url": "https://base-api-notice-management-
api.app.cern.ch/getPolicy/https%3A%2F%2Fglobal-science.org%2Fpolicy",
      "name": "Global Science"
    },
…
```

Per policy request:

```
curl --request GET \
  --url https://base-api-notice-management-
api.app.cern.ch/getPolicy/urn%3Adoi%3A10.60953%2F68611c23-ccc7-4199-96fe-
74a7e6021815
```

Result:

```
{
  "policy": {
```

```
    "augments_policy_uris": null,
    "aut": "https://www.nikhef.nl/",
    "aut_name": "Nikhef",
    "contacts": [
      {
        "email": "helpdesk@nikhef.nl",
        "type": "standard"
      },
      {
        "email": "information-security@nikhef.nl",
        "type": "standard"
      },
      {
        "email": "abuse@nikhef.nl",
        "type": "security"
      },
      {
        "email": "privacy@nikhef.nl",
        "type": "privacy"
      }
    ],
    "description": "This Acceptable Use Policy governs the use of the
Nikhef networking and computer services; all users of these services are
expected to understand and comply to these rules.",
    "description#nl_NL": "Deze Gebruiksvoorwaarden betreffen het gebruik
van netwerk en computers bij Nikhef. Iedere gebruiker van deze middelen of
diensten wordt geacht op hoogte te zijn van deze voorwaarden en deze na te
leven.",
    "id": "urn:doi:10.60953/68611c23-ccc7-4199-96fe-74a7e6021815",
    "includes_policy_uris": [
      "https://documents.egi.eu/document/2623"
    ],
    "notice_refresh_period": 34214400,
    "policy_class": "acceptable-use",
    "policy_url": "https://www.nikhef.nl/aup/",
    "ttl": 604800,
    "valid_from": "2022-04-04T00:00:00Z"
  }
}
```

Annex C: AARC-G100 OID-Federation Pilot: Deployment and Execution Details

This annex provides supporting detail for Section 2.3 and documents the technical setup, participating components, and test scenarios executed as part of the AARC-G100 OpenID Federation pilot.

Pilot Architecture and Entities

The pilot infrastructure consisted of two Trust Authorities, one configured for the Basic Trust Model (G100.1) and one for the Fine-Grained Trust Model (G100.2). Participating SP-IdP-Proxies published entity configurations and registered with one or both Trust Authorities. A Trust Mark Issuer was deployed for the G100.2 profile to support Sirtfi v1/v2 Trust Marks.

Figure 21 illustrates the pilot topology as visualised using the OpenID Federation Browser.

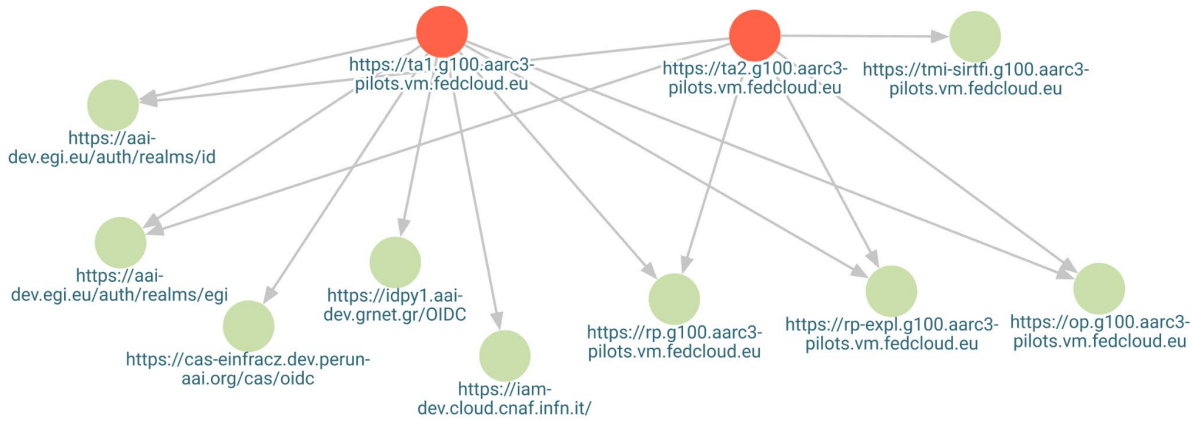

Figure 21 — OpenID Federation (AARC-G100) pilot topology

The participating entities are listed below:

Entity Type	Role in Pilot
Trust Authority 1	Basic Trust Model (G100.1) anchor
Trust Authority 2	Fine-Grained Trust Model (G100.2) anchor
Trust Mark Issuer	Sirtfi Trust Mark issuer for G100.2
Test OP	Validating RP role of proxies (explicit client registration)
Test RP 1	Validating OP role of proxies (explicit client registration)
Test RP 2	Validating OP role of proxies (automatic client registration)
Participating Proxies	Proxies validating AARC-G100 Trust Profiles, based on GÉANT Core AAI

	Platform, Indigo-IAM, Perun AAI (LS AAI), and RCIAM Keycloak (EGI Check-in — e.g. SeaDataNet)

Table C.1: OpenID Federation (AARC-G100) pilot topology

Pilot infrastructure configuration is available in the aarc3-g100-pilot GitHub repository.

Trust Model Scope

AARG-G100 Trust Profile	Description	Required Features
G100.1 – Basic Trust Model	Trust established via federation membership and trust-chain resolution	Entity Config + TA registration + OP/RP flows
G100.2 – Fine-Grained Trust Model	Trust established via federation membership and trust-chain resolution plus enforcement of federation metadata and policy Trust Marks	G100.1 + Trust Mark support + metadata-policy enforcement

Software Support Status

At the time of validation, OpenID Federation adoption across RI-AAI software stacks was still emerging. Four proxy implementations were available for pilot testing, with additional stacks publicly announcing future support.

Support observed:

- G100.1 features implemented to varying degrees across proxies
  - Metadata publishing in both OP and RP roles is available across all implementations.
  - Explicit registration with the proxy acting as OP has been demonstrated successfully by all implementations (each OP has at least one RP that can register and complete a login), although some OP–RP combinations still fail.
  - Explicit registration with the proxy acting as RP has been validated for a subset of implementations; work on the remaining ones is ongoing.
  - Automatic registration with the proxy acting as OP is supported by most implementations and has been shown to work end-to-end.
  - Automatic registration with the proxy acting as RP has not yet been implemented.
- G100.2 Trust Marks + metadata policies not yet implemented

Annex D: Validator Requirements

This annex captures the requirements established in WP4, which guided the design and implementation of the Validator Suite (see Section 3).

[RE1] The validator suite MUST be available as an online service.

[RE2] The validator suite MAY provide an offline way for RIs to perform assessments locally.

[RE3] The validator suite MUST verify the correctness and completeness of the implementation of AARC guidelines, either automated or by self-assessment.

[RE4] The validator suite MUST provide a manual way to self-assess compliance with guidelines that cannot be automatically validated, in the form of a questionnaire; the questionnaire MAY request additional information to be able to prove compliance.

[RE5] The validator suite MUST provide an automated way to evaluate compliance with guidelines where it is possible to automate the process. There are two possible approaches for automatic validation: on-demand vs. periodic.

[RE6] The validator suite MUST implement an on-demand validation approach, which will trigger a log-in of the user requesting the validation every time the validation is requested.

[RE7] The validator suite MAY implement a periodic validation approach, which might impose additional requirements on the RIs.

[RE8] For the periodic validation, the validator suite MUST provide a way for RIs to register for the automatic validation.

[RE9] For the periodic validation, upon registration, the validator suite MAY allow the selection of only specific guidelines for validation.

[RE10] The validator suite implementation MUST allow the addition of validators for new guidelines as they are developed and adopted.

[RE11] The assessment results MUST be visible only to the user who performed the validation.

[RE12] The validator suite MUST provide a solution for publishing the assessment results.

[RE13] The validator suite MAY provide an API to export validation results in a machine-readable way.

[RE14] The validator suite SHOULD provide the RIs with information outlining the necessary steps to achieve compliance with the guideline in cases where they have not met certain requirements.

[RE15] The validator suite MAY provide an API to perform the on-demand automated validation.

Annex E: Validator Example Assessments

This annex presents example assessments generated using the AARC instance of CAT deployed in WP4. These illustrate compliance evaluation for AARC guidelines modelled in the Validator Suite. AARC-G056 and AARC-G069 include automated tests, while AARC-G071 is implemented as a structured self-assessment questionnaire for requirements that cannot be programmatically validated.

AARC-G069 — Group and Role Information

This guideline focuses on a single claim (`entitlements`), which MUST be released via the OIDC UserInfo endpoint and MAY additionally be available via Token Introspection. CAT validates that released entitlements follow the URN schema defined in AARC-G069, and the test is considered successful only if at least one entitlement value matches the required URN format. Figure 22 shows example automated assessment for AARC-G069 specification for expressing group and role information

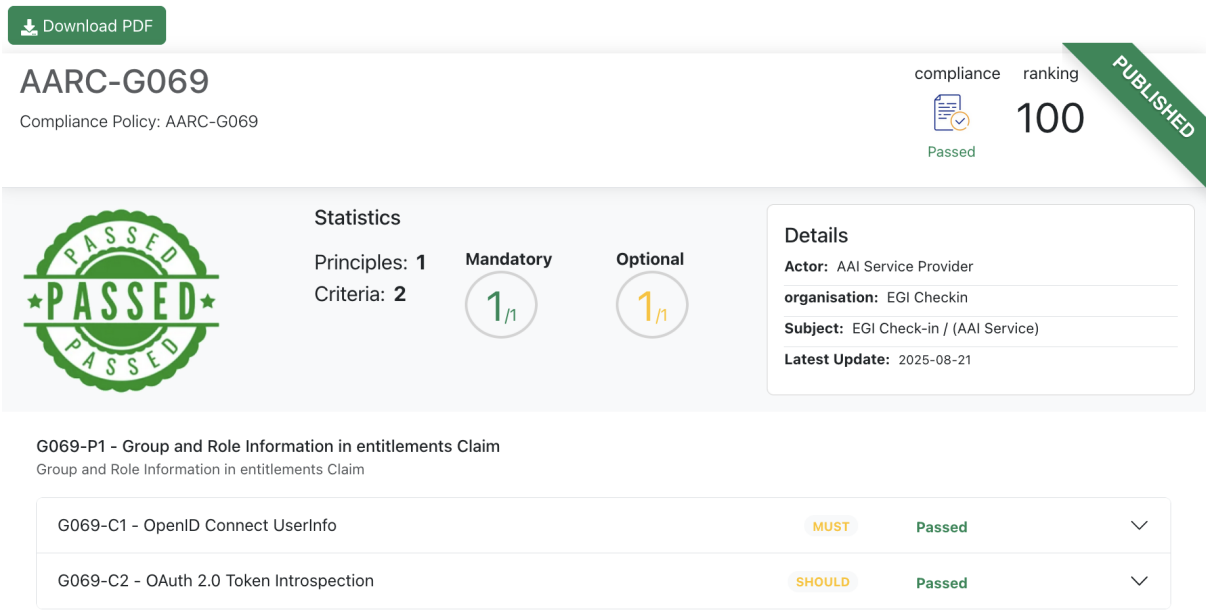

Figure 22 — Example automated assessment for AARC-G069 specification for expressing group and role information

AARC-G056 — Identity Attribute Profile

This guideline evaluates a broader set of identity attributes, including public subject identifiers, email, name, organisation information and assurance attributes, and also checks attribute presence in Access Tokens based on the claim-specific rules defined in the AARC-G056 profile. Figure 23 shows example automated assessment for AARC-G056 Attribute Profile.

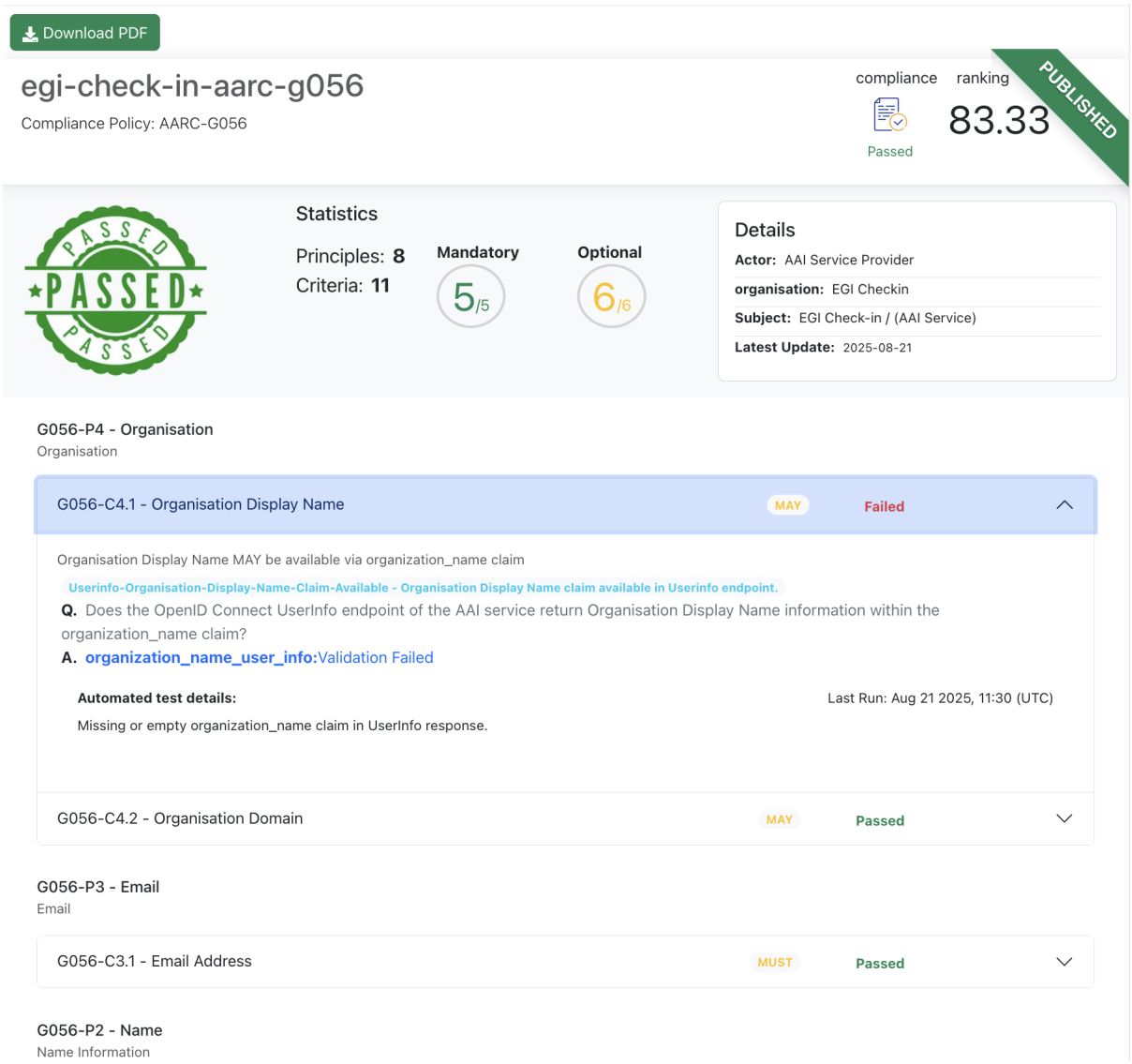

Figure 23 — Example automated assessment for AARC-G056 Attribute Profile

AARC-G071 — Secure Operation of Attribute Authorities (Self-Assessment)

AARC-G071 was modelled in CAT using self-assessment questions rather than automated tests, as the guideline focuses on operational controls, governance processes and trust practices which cannot be validated through protocol inspection alone. Figure 24 shows an example of self-assessment questions for AARC-G071 Guidelines for Secure Operation of Attribute Authorities.

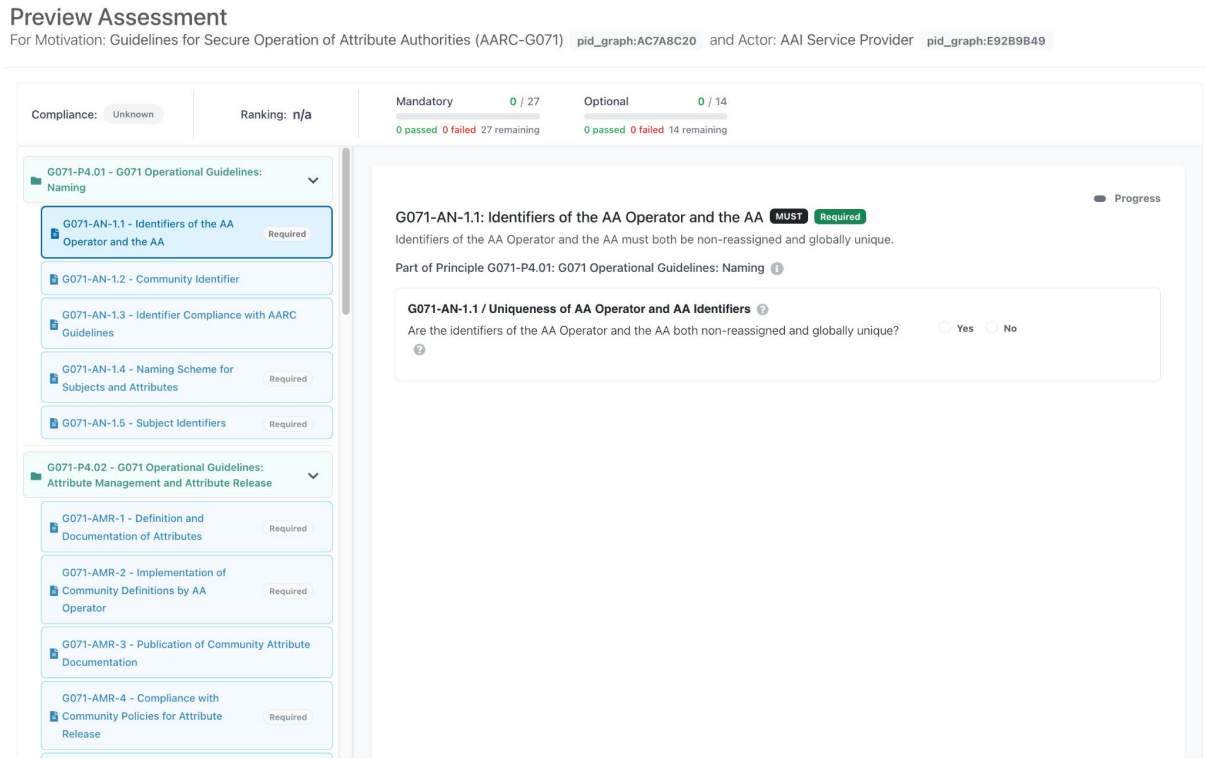

Figure 24 — Example self-assessment questions for AARC-G071 Guidelines for Secure Operation of Attribute Authorities