A specification for IdP hinting

Publication Date	2021-03-10
Authors:	AARC Community members; AppInt members; Marcus Hardt (ed.)
Document Code:	AARC-G061
DOI:	*10.5281/zenodo.4596667*

The work leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme and other sources.

Abstract

This document defines a generic browser-based protocol for conveying - to services - hints about the IdPs or SP-IdP-proxies that should be used for authenticating the principal. This protocol, colloquially referred to as Identity Provider (IdP) hinting, can greatly simplify the discovery process for the end-user, by enabling entities to produce and send hints that can be consumed by SP-IdP-proxies for routing the user to the correct upstream SP-IdP-Proxy or authenticating IdP.

Status of this document

This specification introduces a new hint parameter `aarc_idp_hint` (in Section 3) and obsoletes the specification of the *single-valued* `idphint` parameter [AARC-G049].

Specifically, hint producers can use the `aarc_idp_hint` parameter to specify a single IdP to which the hint consumer should send the user for authentication. The `idphint` parameter on the other hand allows hint producers to specify either a single or multiple IdPs. Functionality of the multi-valued `idphint` parameter is provided by a different parameter [AARC-G064].

A list of all AARC community guidelines and the latest revision of each guideline can be found at https://aarc-community.org/guidelines.

1. Introduction

Authentication at a service in an environment with multiple identity providers (IdPs), requires the service to send users to their IdP. The selection of the IdP is often assisted by IdP Discovery Services, where users can find and select their IdP.

In the AARC Blueprint Architecture (BPA) [AARC-G045], the service that the user tries to access is usually not directly connected to the authenticating IdP (i.e. the IdP identifying the user as in [SAML2Core Sec. 3.4.1.5]), but the connection is through one or more upstream SP-IdP-proxies. The service and each of these SP-IdP-proxies need to be able to route the user authentication request to the IdP interface of the next upstream SP-IdP-Proxy until the authentication request reaches the authenticating IdP. Each time an SP-IdP-Proxy is connected to more than one upstream SP-IdP-proxies or IdPs, the SP-IdP-Proxy will have to rely on an IdP discovery mechanism where the user will need to provide input about the preferred upstream path.

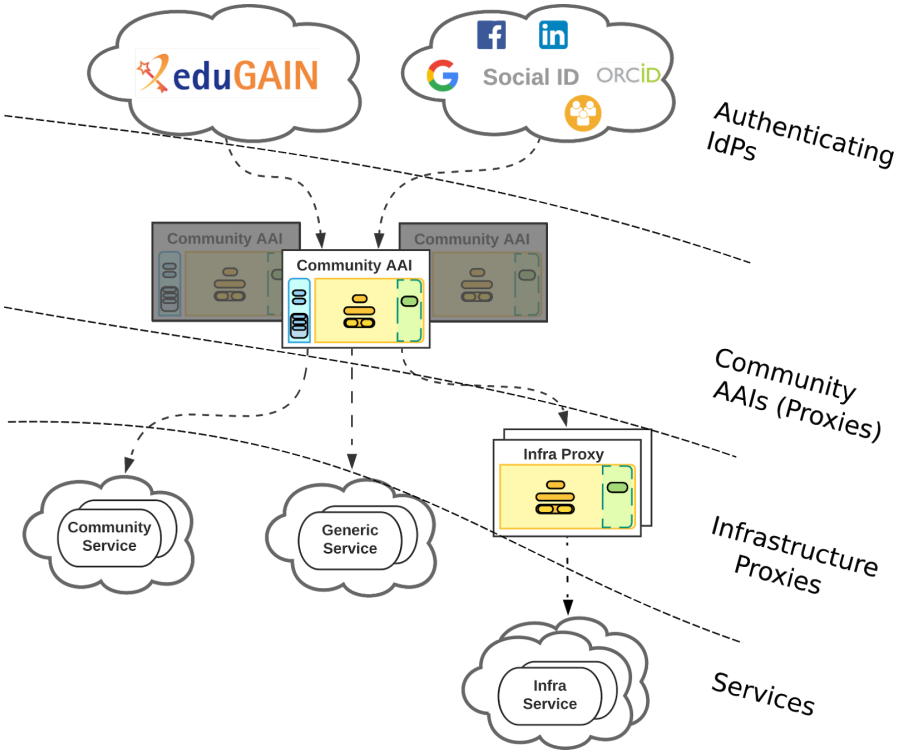

Figure 1: Different entity types shown in the Blueprint Architectures 2019 diagramme

This document defines a portable and technology-agnostic specification that enables entities to produce and send *hints* that can be consumed by SP-IdP-proxies to route the user to the correct upstream SP-IdP-Proxy or IdP, thereby simplifying the discovery process for the end-user.

1.1. Terms and Definitions

The term "Identity Provider" (or "IdP") may refer to SAML IdPs or to OpenID Connect (OIDC) Providers (OPs). Similarly, the term "Service Provider" (or "SP"), may refer to SAML SPs or to OpenID Connect Relying Parties (RPs), and the term “entityID” may refer to SAML entities or to OpenID Connect Providers (OPs).

With a Discovery Service (DS) we mean the logical element providing the user with a list of IdPs to choose from. It can be integrated at services or SP-IdP-proxies.

This document also defines the following terms:

- A "hint producer" produces and sends IdP hints. This may be an SP, or an SP-IdP-Proxy.
- A "hint consumer" receives and consumes IdP hints. This is typically an SP-IdP-Proxy.
- An "entity identifier" identifies an entity, which could for example be an SP, an authenticating IdP, an SP-IdP-Proxy or a Discovery Service. The value of the "entity identifier" may contain the issuer of an OP or the SAML entityID[1].
- The “authenticating IdP” is the IdP at the end of the authentication chain at which the user ultimately identifies. In the AARC BPA [AARC-G045], the authentication chain typically contains one or more intermediate SP-IdP-proxies ([SAML2Core] sec. 3.4.1.5), see fig. 1.
- “Community AAIs” [AARC-G045] serve as authenticating IdPs or SP-IdP-Proxies for specific communities.
- “Multi-tenant Community AAIs” [AARC-G045] serve as SP-IdP-Proxies for multiple communities.
- “Discovery Services” (DS) allow users to choose an IdP for authentication.
- “Infrastructure Proxies” [AARC-G045] are used to connect all services of a specific infrastructure. Infrastructure Proxies may be connected to multiple Community AAIs.
- “Community services” [AARC-G045] are connected to a single Community AAI.
- “Infrastructure services” [AARC-G045] are connected to a single Infrastructure Proxy.
- “Generic services” [AARC-G045] may be connected to multiple Infrastructure- and/or Community AAIs.

1.2. Use Cases

The main use-case of the hints defined in this document is to allow bypassing all or part of the discovery steps: An SP-IdP-Proxy or an SP needs to simplify or bypass the IdP discovery process and route the user directly to the correct IdP. Example use cases include (but are not limited to) the following:

1. A generic or infrastructure service needs to route the user to a specific Community AAI. The service can generate a hint to guide the user directly to this Community AAI (potentially via one or more SP-IdP-Proxies).
2. A service needs the user to re-authenticate at a specific IdP:
    a. to assist the identity linking process; for example, to route the user to the additional IdP that needs to be linked to their identity;

[1] See appendix A for some examples for SAML and OpenID Connect.

   b. to obtain fresh user attribute(s) from the IdP used for the established authentication session;
   c. to obtain fresh user attribute(s) from a different IdP than the one used for the established authentication session at the SP-IdP-Proxy;
3. A community-specific service directory contains URLs to each listed service. Those URLs may already contain a hint pointing at the correct community AAI.
4. An end-service needs to build a simple customised discovery page to replace discovery at the first proxy in the chain that would otherwise show a discovery page.

1.3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Context

- This specification focuses on web-SSO-based flows (both SAML and OIDC), but it does not exclude non-web-browser based flows, nor does it exclude other protocols.
- A trust relation between SPs and IdPs is assumed, but establishing and maintaining it is outside the scope of this document.

3. Specification

The hint parameters[2] specified in this document are:

3.1. `aarc_idp_hint` parameter

1. The value of the `aarc_idp_hint` parameter MUST be a single URL-encoded StringOrURI (section 2 in [RFC7519]). This hint is consumed by an SP-IdP-Proxy.
2. The URL-decoded value of the `aarc_idp_hint` contains at least an entity identifier.
3. The entity identifier MUST identify an SP-IdP-Proxy or authenticating IdP.
4. If the entity identifier is a URI, the URL-decoded value of the `aarc_idp_hint` MAY contain additional query parameters encoding one or more nested hint parameters.

3.2. `idphint` parameter

5. The `idphint` parameter [AARC-G049] is deprecated by a combination of the `aarc_idp_hint` parameter (3.1) and the `aarc_discovery_hint` [AARC-G064].
6. The value of the `idphint` parameter MUST be either
    a. a single URL-encoded StringOrURI providing a hint about a specific SP-IdP-Proxy or specific authenticating IdP. This hint is consumed by an SP-IdP-Proxy; or
    b. a comma-separated list of URL-encoded StringOrURIs providing hints about the SP-IdP-Proxies and/or authenticating IdPs that should be shown to a user. This hint is consumed by a DS.
7. Each URL-decoded StringOrURI contains at least an entity identifier.
8. The entity identifier MUST identify an SP-IdP-Proxy or authenticating IdP.
9. Each URL-decoded StringOrURI that is a URI, MAY contain additional query parameters encoding one or more nested hint parameters.

3.3. General rules

These general rules apply to both the `aarc_idp_hint` and to the `idphint` parameters.

10. The hint consumer MUST be capable of processing each of the hint parameters in HTTP GET requests.
11. The hint consumer SHOULD be capable of processing each of the hint parameters in HTTP POST requests.

[2] The inconsistent naming of the `idphint` parameter is necessary to retain backwards compatibility.

12. Handling hints:
    a. A hint consumer MAY process a hint by ignoring all or part of the value of each received hint parameter.
    b. A hint consumer MAY process a hint by forwarding it to another hint consumer instead of, or in addition to, fulfilling it itself[3].
13. Handling nested hints:
    a. A hint producer MAY produce hints carrying nested hint parameters.
    b. A hint consumer MAY ignore any nested hint parameter.
    c. If nested hints are added to a hint, they MUST be added to the entity identifier's URI in the form of query parameters.
    d. If a hint consumer handles a nested hint, the consumer MUST send this nested hint using a protocol understood by the next consumer. This means that it may need to use a different protocol or mechanism than the one through which the original hint was received.
14. URIs MUST be URL-encoded when used as values of hints with the following additional rules:
    a. Forward slashes ('/'), and commas (',') MUST be percent-encoded ([RFC3986] section-2.1).
    b. Case sensitivity of the encoded value MUST follow the underlying specification of the original unencoded value.
15. To obtain the entity identifier from an incoming URI value, the hint consumer MUST remove all of the following parameters where applicable[4]:
    a. the `idphint` parameter and
    b. parameters prefixed with `aarc_`

    Within the scope of this specification what is left is assumed to be the entity identifier (*including* any remaining query parameters).

[3] For example, a consumer which is connected to only one IdP and receives a hint for a different IdP than the one it is connected to, MAY forward the hint to its connected IdP.

[4] Some entityIds contain query parameters. Future revisions of this specification may also introduce new hint parameters. Therefore, these new parameters will be prefixed with "aarc_" in order to avoid namespace collision with other parameters.

4. Implementation Considerations

4.1. Mandatory vs. Optional to Implement Features

- The `aarc_idp_hint` deprecates the *single-valued* `idphint` parameter[5], specified in [AARC-G049].
- Support for the `aarc_idp_hint` is mandatory for hint producers and hint consumers implementing this specification.
- The `idphint` parameter MUST at least be supported such that hint consumers will
    a. not fail upon receiving the `idphint` parameter; and
    b. follow rule number 15 a. when obtaining the entity identifier
- Hint producers that can produce both `idphint` and `aarc_idp_hint` parameters SHOULD prefer the `aarc_idp_hint` parameter.
- Implementers of this specification are encouraged to support both the `aarc_idp_hint` and the `idphint` parameters in order to retain backwards compatibility with the previous version of this specification.
- When producers send both an `aarc_idp_hint` and a *single-valued*[5] `idphint` parameter the value of the two parameters MUST be the same.
- When consumers receive both an `aarc_idp_hint` and a *single-valued*[5] `idphint` parameter the consumer SHOULD ignore the value of the `idphint` parameter.

4.2. Processing Rules for consumers of nested hints in `aarc_idp_hint`

The general form of an `aarc_idp_hint` parameter is[6]:

```
aarc_idp_hint=urlencode(<entityID[?<nested_hint>[&<nested_hint>]...]>)
```

where each optional `<nested_hint>` request parameter is of the form:

```
<idphint|aarc_*>=urlencode(<entityID[?<nested_hint>[&<nested_hint>]...]>)
```

To process a hint with a value that contains a nested hint correctly, it is necessary to

1. Extract the actual entity identifier from the value of the hint parameter (point 15)
2. Decide whether to use the hint (based on the entity identifier).
3. Re-add the nested hint to the URI to which the user is forwarded (point 13 d).

A concrete example is given in Appendix A.2.

[5] Functionality of the multi-valued idphint parameter is provided by a different parameter[AARC-G064].
[6] In case the entityID itself contains a query parameter & is used instead of the ?.

References

[AARC-G045]	AARC Blueprint Architecture-2019 https://aarc-community.org/guidelines/aarc-g045
[AARC-G049]	A specification for IdP hinting https://aarc-community.org/guidelines/aarc-g049
[AARC-G064]	A specification for passing hints to Discovery Services (to be published) https://aarc-community.org/guidelines/aarc-g064
[IDPDSP]	SAML Identity Provider Discovery Service Protocol and Profile http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery.pdf
[OIDC-Core]	OpenID Connect 1.0 Specification https://openid.net/specs/openid-connect-core-1_0.html
[OIDF-PL]	OpenID Connect Federation 1.0, Section 4: Policy Languagle https://openid.net/specs/openid-connect-federation-1_0.html#rfc.section.4
[RFC2119]	Key words for use in RFCs to indicate Requirement levels https://tools.ietf.org/html/rfc2119
[RFC3986]	Uniform Resource Identifier (URI): Generic Syntax https://tools.ietf.org/html/rfc3986
[RFC7519]	JSON Web Token (JWT) https://tools.ietf.org/html/rfc7519
[SAML2Core]	SAML2-Core-OS http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
[REFEDS-RNS]	Research and Scholarship Entity Category https://refeds.org/category/research-and-scholarship
[REFEDS-SIRTFI]	Security Incident Response Trust Framework for Federated Identity https://refeds.org/sirtfi

Appendix A: Examples

The examples shown contain the actual links to be used for hinting. For clarity, we use the pseudocode that generates the URI. Backslash-escaped newlines are inserted for readability purposes only.

Note that the mechanism is independent of the protocol of the hinted IdP. We make use of SAML nomenclature and entityIDs, but any expression can also be used in an equivalent way in OpenID Connect.

In the following examples, `https://community-aai.org` is the SAML entityID for a Community AAI SP-IdP proxy. Its corresponding SAML SSO HTTP-Redirect location is `https://community-aai.org/idp/saml/SSO`. Likewise, `https://home-idp.edu` is the SAML entityID for an authenticating IdP.
We also use an Infrastructure AAI SP-IdP proxy with SAML SSO HTTP-Redirect location `https://infra-proxy.org/idp/saml/SSO` and for its `OIDC` authorization endpoint `https://infra-proxy.org/authorize`.

The following are real life examples of SAML entityIDs:

- `urn:mace:kuleuven.be:kulassoc:kuleuven.be`
- `urn:idp:cnam`
- `https://idp.scc.kit.edu/idp/shibboleth`
- `https://accounts.google.com/o/saml2?idpid=C02a9w`
- `https://proxy.eduteams.org/proxy`

The following are real life examples of OAuth2 AS or OIDC OP issuers:

- `https://aai.egi.eu/oidc/`
- `https://accounts.google.com`
- `https://b2access.eudat.eu/oauth2`
- `https://cilogon.org`

A.1 Basic hinting

Example 1. A community service produces a hint to inform the community AAI proxy at `community-aai.org` that the user should be authenticated at the hinted IdP (`home-idp.edu`). A non-normative example of the URL to which the service redirects the users browser is then:

```
https://community-aai.org/idp/saml/SSO
      ?SAMLRequest=...
      &aarc_idp_hint=$(urlencode(https://home-idp.edu))
```

Upon receiving the URL, the community proxy decodes the hint and should then directly redirect the user to `home-idp.edu` to authenticate (provided that the community proxy trusts the identified home IdP and allows it for that community service).

Example 2. In case the community service needs to send the hint using the older version of this specification, the URL needs to be extended with an `idphint` parameter with the same value:

```
https://community-aai.org/idp/saml/SSO
      ?SAMLRequest=...
      &aarc_idp_hint=$(urlencode(https://home-idp.edu))
      &idphint=$(urlencode(https://home-idp.edu))
```

Example 3. An infrastructure service produces a hint to inform the infrastructure proxy that the user should be redirected to the hinted community AAI. A non-normative example of the URL to which the service redirects the users browser is then:

```
https://infra-proxy.org/idp/saml/SSO
      ?SAMLRequest=...
      &aarc_idp_hint=$(urlencode(https://community-aai.org))
```

Example 4. An OIDC based infrastructure service (OIDC client) produces a hint to inform the infrastructure proxy (OIDC OP) that the user should be redirected to the hinted community AAI. A non-normative example of the URL to which the service redirects the users browser is then:

```
https://infra-proxy.org/authorize
      ?scope=openid
      &response_type=code
      &client_id=foobar
      &redirect_uri=$(urlencode(https://client.example.org))
      &state=...
      &aarc_idp_hint=$(urlencode(https://community-aai.org))
```

A.2 Nested hinting

Example 1. An infrastructure service produces a hint to inform the infrastructure proxy that the user should be authenticated at the hinted community AAI, which in turn is hinted to redirect the user to a specific authenticating IdP. This may be necessary because the infrastructure proxy can be connected to several Community AAIs. A non-normative example of the URL to which the service redirects the users browser is then something like:

```
https://infra-proxy.org/idp/saml/SSO
      ?SAMLRequest=...
      &aarc_idp_hint=$(urlencode(https://community-aai.org
```

```
            ?aarc_idp_hint=$(urlencode(https://home-idp.edu))
            &idphint=$(urlencode(https://home-idp.edu))
      ))
```

The infrastructure proxy would then redirect the user to:

```
https://community-aai.org/idp/saml/SSO
      ?SAMLRequest=...
      &aarc_idp_hint=$(urlencode(https://home-idp.edu))
      &idphint=$(urlencode(https://home-idp.edu))
```

Example 2. Similar to example 1 but for OpenID Connect. Here the hint would typically be attached as an extra request parameter to the OAuth 2.0 authorisation request, used by the client to redirect the user to:

```
https://infra-proxy.org/authorize
      ?scope=openid
      &response_type=code
      &client_id=foobar
      &redirect_uri=$(urlencode(https://client.example.org/cb))
      &state=...
      &aarc_idp_hint=$(urlencode(https://community-aai.org
            ?aarc_idp_hint=$(urlencode(https://home-idp.edu))
            &idphint=$(urlencode(https://home-idp.edu))
      ))
```

A.3 Non normative examples

Hinting as defined in this document may be used in situations that are not directly targeted by this document.

Services as hint consumers

A service may consume hints. This is particularly useful in the case of generic services that support multiple IdPs (e.g. a set of SP-IdP-proxiesand/or authenticating IdPs). Links to that service (e.g. as listed in a VO directory) may then make use of hints specified in this document, such as:

```
https://generic.service.edu/
      ?aarc_idp_hint=$(urlencode(https://einfra-aai.org/idp/saml))
```

Real life example: Science Gateway using RCauth CA

In the RCauth scenario VO portals (Science Gateways) are connected to a MasterPortal which acts as intermediary between the VO portal and the RCauth Online CA. When one of the VO portals sends the user to login at the MasterPortal, the latter sends the user further on to the RCauth CA itself where the user will normally be presented with a discovery page. In this login process, VO portals can send an IdP hint to the MasterPortal, but the hint will always be forwarded to the RCauth CA. The MasterPortal in this scenario is therefore acting as an SP-IdP proxy (on a community or infrastructure level) but connected to a single IdP only (namely the RCauth CA).