Document was moved here:

https://docs.google.com/document/d/1iG9R6oXjVnJx-3yH4hFThcb9AuXkR8hIDNbncmHJ0nI/edit

(in hope of less bugs and better speed)

A specification for IdP hinting

Publication Date 2019-02-27
Authors: AARC Community members;AppInt members;Marcus Hardt (ed.)

Document Code: AARC-G049
DOI:

The research leading to these results has received funding from the European Community's Horizon2020 Programme under Grant Agreement No. 730941 (AARC2).

Abstract

This document defines a generic browser-based protocol for conveying - to services - hints about the IdPs or IdP-SP-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 either narrowing down the number of possible/IdPs to choose from or by making the actual selection process fully transparent.

Table of Contents

1. Introduction

This document aims to be technology agnostic. Whenever we use the term “Identity Provider” (or “IdP”) we refer to SAML IdPs or to OpenID Connect Providers (OPs). Similarly, when we use the term “Service Provider” (or “SP”), we refer to SAML SPs or to OpenID Connect Relying Parties (RPs).

Authentication at a service in a multi-IdP environment requires the service to redirect the incoming user to their home identity provider. Currently this is often accomplished by Discovery Services, where users choose their home IdP.

The rise of the proxy concept introduces one or more IdP interfaces between the service and home IdP of the user. In cases where a service might be connected to more than one IdP-SP-Proxy, users have to choose from a list of IdP-SP-Proxies. This makes it increasingly difficult for users to understand which IdP is the best choice for authentication.

In this document we focus on enabling Service Providers to create or forward *hints* that can be used by IdP-SP-Proxies or Discovery Services in order to drive the IdP Discovery Selection process. We define a portable and technology-agnostic way to allow services to provide hints for proxies and/or Discovery Services, which can be used by a Proxy to choose the IdP to redirect the user to, or by a Discovery Service to narrow down the list of Identity Providers that should be presented to the users. These hints can be added at multiple points in the flow and range from specifying static URLs to adding the hint dynamically during the authentication flow.

This mechanism can greatly simplify the discovery process for the end-user, by either narrowing down the number of possible IdPs to choose from or by making the actual selection process fully transparent.

Additionally it provides a way of “branding” a single service for different communities, by providing different links to their users or by directing the user to community branded discovery services.

Furthermore, the described concept includes the possibility of chaining by creating nested hints. This allows creating URLs that point to an SP, with a hint trail that leads via one or more IdP-SP-Proxies to for example a given home IdP.

Finally, we want to stress that this hinting process is independent of the underlying protocol used. The hints themselves, however, may contain protocol specific information. We also stress that they are only hints: Whether the Proxy, Service Provider or Discovery Service actually honours them is depending on their local policies.

1.1. 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

For the scope of this document, we define these abstract entities:

- A “hint producer”, that produces hints. This may for example be an SP, or an AARC BPA proxy.
- A “hint consumer”, that receives hints. This may for example be an SP, an IdP, an AARC BPA proxy or a Discovery Service.
- An “entity identifier” identifies an entity. This entity may for example be an SP, an IdP, an AARC BPA proxy or a Discovery Service.

The hinting mechanism described in this document is based on the following assumptions:

- This specification assumes web-SSO-based flows (both SAML and OIDC), but it does not exclude non-web-browser based flows, nor does it exclude other protocols.
- The trust relation between SPs and IdPs is outside the scope of this document.
- This specification supports, but by no means requires, that services are operated in an AARC BPA context.
- The hint consumer can either itself process the hint, or decide to pass the hint to another hint consumer such as its Discovery Service for filtering the list of potential IdPs. Details for this are out of the scope of this document.
- Hint producers that want to point users to a specific home-IdP for reauthentication, will need to know the necessary identifier (such as a SAML entityID or OIDC issuer) to do so.

Hints can be used, for example, by an SP when redirecting a user to an IdP-SP-Proxy to pass additional information to that IdP-SP-Proxy. This information may include pieces of information that the given IdP-SP-Proxy should pass on to the next entity, when redirecting a user there.

3. Specification

In this section we first define the different types of hints introduced in this specification. We then provide the rules which are common across all of them, followed by the rules specific for each of the different types of hints separately, if applicable. Hint parameters specified in this document:

- `idphint:` a hint about which **IdP** to use
- `ds_idps_hint:` a hint about which **IdPs** to show, consumed by the DS
- `ds_hint:` a hint about which **DS** to use
- `sp_origin:` allows a proxy to indicate the originating **SP**

3.1. General rules

1. The hint consumer MUST be capable of processing each of the hint parameters in GET requests.

2. The hint consumer SHOULD be capable of processing each of the hint parameters in POST requests.
3. Each URI in the value of a hint parameter MUST consist of a URN or URL indicating an entity identifier, optionally extended with another hint parameter (chained hint).
4. URIs MUST be URL-encoded when used in values of hints.
    a. Forward slashes ('/') MUST be percent-encoded [https://tools.ietf.org/html/rfc3986#section-2.1].
    b. Case sensitivity MUST follow the underlying specification of the corresponding URL-decoded value.
5. When receiving a chained hint, the hint consumer SHOULD send the nested hint (i.e. the next level hint) using a protocol understood by the next service (i.e. the first level hint) in the chain. It MAY use a different protocol or mechanism than the one through which it received the chained hint.
6. A hint consumer MAY ignore all or part of the value of each received hint parameter[1].

3.2. Specific rules

3.2.1. idphint parameter

1. The `idphint` value MUST be a single URL-encoded URI.
2. The entity identifier in an `idphint` MUST identify an IdP.
3. [20.] The `idphint` parameter MUST not be combined with the `ds_idps_hint` and/or `dshint` parameters in the same request.
4. [8.] The hint producer SHOULD NOT combine the `idphint` parameter with other hinting mechanisms not defined by this spec (e.g. SAML scoping concept [SAML2CoreSAML-CORE-2.0-OS]).
5. [9.] The hint consumer MUST NOT use the value of a received idphint parameter in combination with hints received via other hinting mechanisms.[2]

3.2.2. ds_idps_hint parameter

6. [10.] The hint producer can use the `ds_idps_hint` to provide a list of IdP identifiers that SHOULD be used by the Discovery Service to filter the list of available IdPs
7. [4.] The value of the `ds_idps_hint` parameter MUST be one or more comma-separated URL-encoded URIs. [RFC3986].
    a.
8. [11. First sentence] The value of the `ds_idplist_hint` parameter MUST be one or more, comma-separated, URL-encoded URIs [RFC3986]. Implementations MUST also URL-encode slashes ('/').

---

[1] For example, in the case of `idphint`, a hint consumer can ignore the value if the hinted IdP is unknown or disallowed for a certain relying party.

[2]For example, if a hint consumer receives both, an `idphint` and an `<IDPList>`idplist [SAML2CoreSAML-CORE-2.0-OS], one of them should take precedence., The consumer MUST NOTbut mergeing, or cross secting the two received hints must not be done.

9. [14.] If the effective list of available IdPs (e.g. by intersecting its own list of trusted IdPs and the received list of hinted IdPs) contains exactly one element, the hint consumer SHOULD immediately redirect the user to this referenced entity.

3.3.3 ds_hint parameter

11. [15.] The hint producer can use the `dshint` to provide an identifier for the Discovery Service that should be used
12. [16.] The value of the `dshint` parameter MUST be one, URL-encoded URIs [RFC3986]. Implementations MUST also URL-encode slashes ('/').
13. [17.] A hint consumer MAY ignore the value of an incoming `dshint` parameter, for example because the hinted Discovery Service is unknown

3.3.4 sp_origin parameter

15. [18.] The hint producer can use the `sporigin` parameter to provide an identifier for the originating SP. This information is very useful for Discovery Services that are connected to proxies as from the point of view of the Discovery Service, the service requesting discovery is the proxy. With this hint, the IdP-SP-Proxy can signal to the Discovery Service, which the originating SP was.
16. [19.] The value of the `sporigin` parameter MUST be one, URL-encoded URI [RFC3986]. Implementations MUST also URL-encode slashes ('/').

4. Parsing Rules

The hint consumer MUST parse the value of the `idphint` (called `hint_1` in the following) as follows:

1. URL-decode `hint_1`, the result is `dec_hint_1`.
2. if `dec_hint_1` itself contains an `idphint` query parameter, remove and store it as hint_2. The shortened `dec_hint_1` will be called `entity_1`.
3. `entity_1` MUST be interpreted as the unique identifier for the IdP. For SAML this is its entityID, for an OIDC Provider or OAuth2 AS this is the issuer.
4. if `hint_2` in step 2. is non-empty, the hint consumer MUST re-add it as `idphint` when it is going to send the user to the IdP or OP specified by `entity_1`.

5. Specification

5.1. *-- or is it?*

5.2. Core

1. The hint consumer MUST be capable of processing the `hint` parameters in GET requests.

2. The hint consumer SHOULD be capable of processing the `hint` parameters in POST requests.
3. The identifier of the hinted IdP MUST be passed using the `idphint` parameter.
4. The value of the `idphint` parameter MUST be one URL-encoded URI [RFC3986].
   a. Implementations MUST also URL-encode slashes ('/').
   b. Case sensitivity MUST follow the underlying specification of the URL-decoded identifier.
5. The URI in the value of the `idphint` parameter MUST consist of a URN or URL identifying an IdP, optionally extended with another `idphint` or a `ds_idplist_hint` query parameter (chained `idphint`).
6. When receiving a chained `idphint`, the hint consumer SHOULD send the nested `idphint` (i.e. the next level hint) using a protocol understood by the next service (i.e. the first level hint) in the chain. It MAY use a different protocol or mechanism than the one through which it received the chained `idphint`.
7. A hint consumer MAY ignore the **value** of an incoming `hint` parameter. For example, in the case of `idphint`, a consumer can ignore the valuee if the hinted IdP is unknown or forbidden for a certain relying party.
8. The hint producer SHOULD NOT combine the `idphint` parameter with other hinting mechanisms (e.g. SAML scoping concept [SAML2CoreSAML-CORE-2.0-OS]).
9. The hint consumer MUST NOT combine the `idphint` parameter with other hinting mechanisms (e.g. SAML scoping concept [SAML2CoreSAML-CORE-2.0-OS]). Which mechanism takes precedence MAY be configurable
10. The hint producer can use the `ds_idplist_hint` to provide a list of IdP identifiers that SHOULD be used by the Discovery Service to filter the list of available IdPs
11. The value of the `ds_idplist_hint` parameter MUST be one or more, comma-separated, URL-encoded URIs [RFC3986]. Implementations MUST also URL-encode slashes ('/').
12. Each URI included in the value of the `ds_idplist_hint` parameter MUST consist of a URN or URL identifying an IdP
13. A hint consumer MAY ignore all or part of the **value** of an incoming `ds_idplist_hint` parameter, for example because the hinted IdPs are unknown or perhaps it disallows a certain IdP for a certain relying party.
14. If the effective list of available IdPs (e.g. by intersecting its own list of trusted IdPs and the received list of hinted IdPs) contains exactly one element, the hint consumer SHOULD immediately redirect the user to this referenced entity.
15. The hint producer can use the `dshint` to provide an identifier for the Discovery Service that should be used
16. The value of the `dshint` parameter MUST be one, URL-encoded URIs [RFC3986]. Implementations MUST also URL-encode slashes ('/').
17. A hint consumer MAY ignore the **value** of an incoming `dshint` parameter, for example because the hinted Discovery Service is unknown
18. The hint producer can use the `sporigin` parameter to provide an identifier for the originating SP. This information is very useful for Discovery Services that are connected to proxies as from the point of view of the Discovery Service, the service

requesting discovery is the proxy. With this hint, the IdP-SP-Proxy can signal to the Discovery Service, which the originating SP was.

19. The value of the `sporigin` parameter MUST be one, URL-encoded URI [RFC3986]. Implementations MUST also URL-encode slashes ('/').
20. If the hint producer provided an idphint parameter then ds_idplist_hint and/or dshint parameters MUST not be added in the request

3.2 Parsing Rules

The hint consumer MUST parse the value of the `idphint` (called `hint_1` in the following) as follows:

5. URL-decode `hint_1`, the result is `dec_hint_1`.
6. if `dec_hint_1` itself contains an `idphint` query parameter, remove and store it as hint_2. The shortened `dec_hint_1` will be called `entity_1`.
7. `entity_1` MUST be interpreted as the entityIDof a SAML IdP or issuer of an OIDC Provider or OAuth2 AS.
8. if `hint_2` in step 2. is non-empty, the hint consumer MUST re-add it as `idphint` when it is going to send the user to the IdP or OP specified by `entity_1`.

References

[BPA2019] AARC Blueprint Architecture-2019
https://aarc-project.eu/guidelines/aarc-g045

[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

[SAML2Core] SAML2-Core-OS §3.4.1.2
http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Appendix A: Examples

The examples shown contain the actual links to be used for hinting. For clarity, we also provide the pseudocode that generated the link).

Note that the mechanism is independent of the protocol of the hinted IdP. We make use of this in the second examples, where we first hint to an OAuth2 endpoint at the proxy and then to a SAML endpoint of the home-IdP identified by its entityID.

Examples for authenting entities

Examples for SAML EntityIDs:

- urn:mace:kuleuven.be:kulassoc:kuleuven.be
- urn:idp:cnam
- https://idp.scc.kit.edu/idp/shibboleth

Examples for OAuth2 ASs or OIDC OPs:

- https://b2access.eudat.eu/oauth2
- https://accounts.google.com

Simple example

`https://example.service.edu/?idphint=https%3A%2F%2Fhome-idp.org%2Fidp%2Fsaml`

<=>

`https://example.service.edu/?idphint=`*`urlencode(`***`https://home-idp.org/idp/saml`***`)`**

Service `https://example.service.edu` receives a hint to authenticate the user at SAML entity ID `https://home-idp.org/idp/saml`

Multiple IdP example

`https://example.service.edu/?idphint=urn%3Amace%3Aone-proxy.org,https%3A%2F%2Fanother-proxy.org`

<=>

`https://example.service.edu/?idphint=`*`urlencode(`***`urn:mace:one-proxy.org`***`)`**`,`***`urlencode(`***`https://another-proxy.org`***`)`**

Service https://example.service.edu gets a list of entities (namely `urn:mace:one-proxy.org` and `https://another-proxy.org`) at which the user is suggested to be authenticated.

Chained hinting example

https://example.service.edu/?idphint=https%3A%2F%2Fidp-sp-proxy.org%2Foauth2%3Fidphint%3Dhttps%253A%252F%252Fhome-idp.org%252Fidp%saml

<=>

https://example.service.edu/?idphint=*urlencode(***https://idp-sp-proxy.org/oauth2?idphint=***urlencode(***https://home-idp.org/idp/saml***))**

Service `https://example.service.edu` receives a hint to authenticate the user at the OAuth2 AS with issuer `https://idp-sp-proxy.org/oauth2`, and will redirect the user in the OAuth2 authorize request to the URL[3] `https://idp-sp-proxy.org/oauth2/authorize?idphint=https%3A%2F%2Fhome-idp.org%2Fidp%2Fsaml` thereby passing the URL to the proxy with an encoded idphint parameter to the home-IdP.

The url encoded parameter in turn will hint the proxy to redirect the user to authenticate at `https://home-idp.org/idp/saml`.

Multipath hinting

A service in the context of the “community first” approach, described in the 2019 version of the Blueprint Architecture [BPA-2019] may use this specification to enforce the use of a specific community identity, but supporting multiple infrastructure proxies. It can then hint different proxy-paths to the same community AAI.

[3] Note that the actual `/authorize` endpoint might be different from the one given here, the proxy will generally check the relevant .well-known endpoint or the SAML metadata for obtaining it.