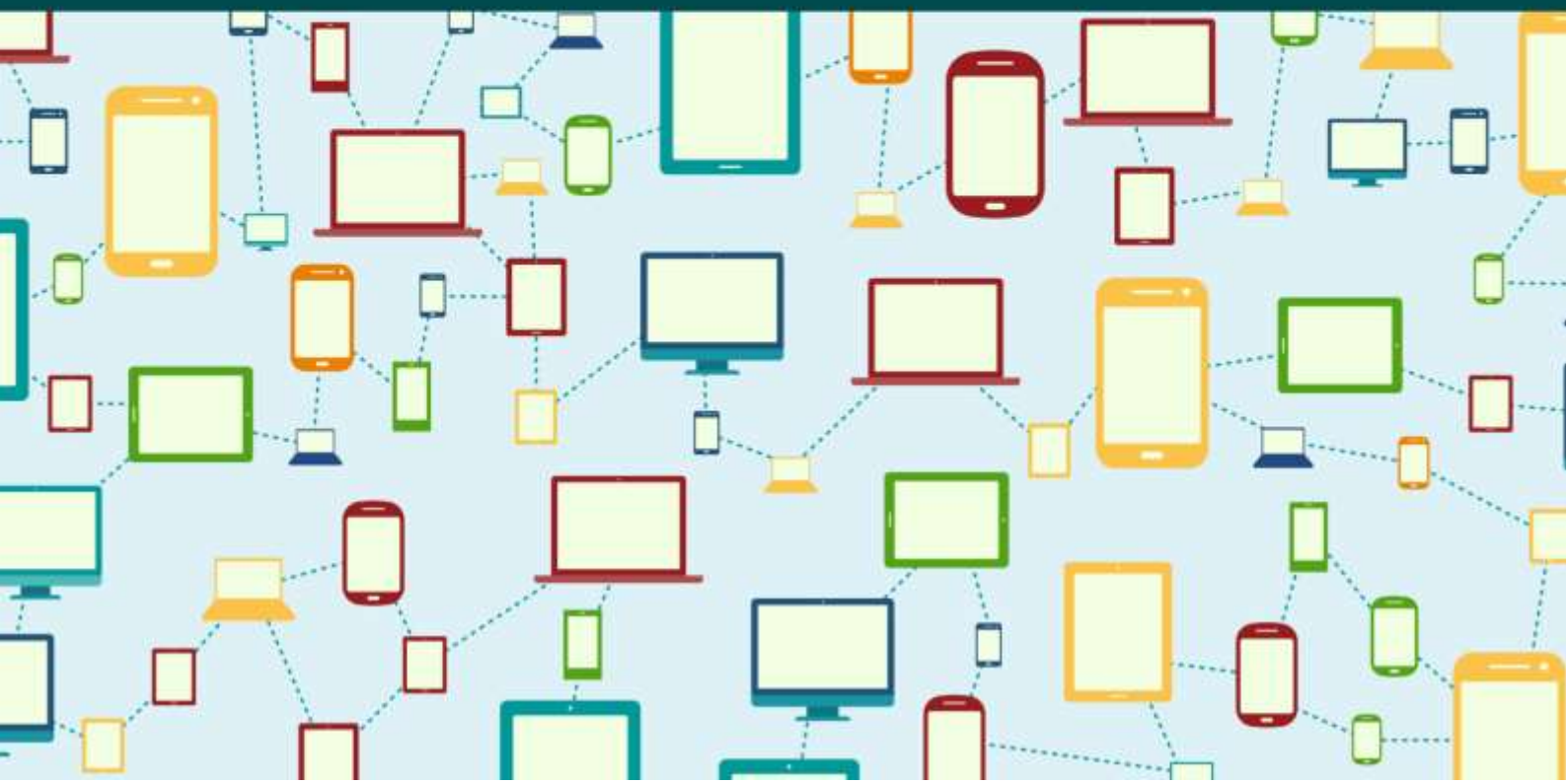
fido alliance | simpler stronger authentication



Enterprise Adoption Best Practices

Integrating FIDO & Federation Protocols

December 2017

Audience

This white paper is aimed at enterprises deploying FIDO for strong authentication. It is intended to provide guidance to architects and developers on how to integrate FIDO authentication and existing federation protocols, namely SAML and OpenID Connect.

It is assumed that the reader has an understanding of FIDO architecture and protocols.

Contents

1 Introduction

Many enterprises have implemented federation protocols, such as SAML and OpenID Connect, within their identity platforms in order to provide an improved user experience to end users, as well as better security for the enterprise. As those enterprises adopt the FIDO authentication protocol they want to understand how to best leverage it in a federated environment. Fortunately these two technologies – federation and FIDO – are complementary. The value of a FIDO authentication capability is amplified by a federated system, where the federation system extends the benefits of a FIDO authentication to applications and services without requiring FIDO to be directly integrated with those applications. The purpose of this document is to share best practices for using FIDO together with federation protocols.

FIDO and federation technology have some things in common. Both seek to simplify the end-user's experience in authenticating to applications provided my multiple providers (referred to as Relying Parties, or RPs in this document). But there are also significant differences between the two – an important one being the trust model that is defined in their scenarios. While FIDO enables a user to register an authentication credential with multiple application/service providers, federation involves an additional actor, known variously as an Identity Provider (IDP) or OpenID Connect Provider (OP). This third party IDP or OP is responsible for authenticating the end user and providing information about the user and the authentication event to multiple service providers. It is in this way that federation 'amplifies' a FIDO deployment – by enabling the user to register their FIDO token with an IDP and then be able to leverage that FIDO token across applications/services without needing to register that token directly with each of those relying parties.

It is also possible that a federation Relying Party could itself perform a FIDO authentication directly, in order to elevate the assurance in the user's identity delivered through the federation flow.

It should be additionally noted that by design, FIDO protocols restrict an authenticator from using the same private key to authenticate to different application domains. Consequently, FIDO is not an appropriate solution for giving users a Single Sign-On (SSO) experience *across* different application domains, which is exactly what federation protocols can enable. FIDO enables an end user to use one device to authenticate to multiple relying parties over the lifecycle of their interaction with those RPs, whereas federated SSO protocols enable the user to leverage one authentication event to access multiple relying parties during individual sessions.

Enterprises provide a variety of types of services to a variety of end users and devices, and the security and user experience requirements vary widely across the different scenarios they support. In order to accommodate this variety of scenarios, enterprises often deploy extensible identity platforms that have the ability to leverage new forms of authentication – allowing the user to use an authentication mechanism that is deemed 'appropriate' for a given scenario. As a result, there are a couple of primary ways that FIDO authenticators are be used by enterprises – (1) to authenticate the user when they first begin to use the service, or (2) to 'step up' the authentication method for a user session to a FIDO-based method (usually when they attempt to access something more sensitive).

The full value of the above integration presumes that the applications can

1. Request of the federation server, either explicitly or implicitly, that the authentication be FIDO-based, or even use a specific FIDO authenticator
2. Be informed of the fact that the initial authentication was FIDO-based, and even the specific FIDO authenticator used.

Federation deployments can differ with respect to how the type of authentication to be performed is determined. In some scenarios the service/application provider will request that the IDP perform a specific type of authentication. In other scenarios, the SP simply asks the IDP to authenticate the end user and the IDP decides what method of authentication to use. Federation protocols generally provide mechanisms by which federation

actors can refer to particular authentication mechanisms - either as part of a request that a user should be authenticated with that mechanism, or an assertion that a user was authenticated in that manner.

When the actual authentication mechanism is FIDO-based, then the means by which authentication mechanisms are referred to in the federation protocol must be translated into an appropriate FIDO authenticator. This document provides guidance on this mapping.

There is more than one way implementing FIDO given that multiple approaches are defined by different parts of the specification set – specifically UAF, U2F, and the FIDO2 protocols. This document attempts to speak to their commonalities where possible, and call out the differences where they are significant to enterprise deployments.

2 Definitions

Please note that some of these definitions are different than the FIDO Glossary's definition of terms, as FIDO has used them in a more specific way than they are often used in federation. We will mark those definitions with a '*' here.

- **Authentication (*)** – a method or process that a user employs to prove their identity to a third party. FIDO authentication represents one such method, and examples of other approaches are username/password, biometric authentication, and One Time Passwords. Enterprises use authentication to ensure that users have the appropriate degree of assurance that the user is who they claim to be as those users access a given application.
- **Federation** - a model of identity management that leverages open standards to share user information across security/application domains to enable users to securely access applications across those security domains. Federation technology enables Authentication, Authorization, and Provisioning information to be shared between security domains in order to enable application access scenarios between those domains. For the purposes of this document we will focus on the use cases involving the sharing of Authentication information – for the purposes of Delegated Authentication and Single Sign-on. There are two primary federated SSO standards that we will discuss in this document: the Security Assertion Markup Language (SAML) and OpenID Connect (OIDC). These standards describe a protocol that enables users to authenticate to one domain and then leverage that authentication to access applications within another domain. Federation abstracts away from the two policy domains the specifics of the identity management infrastructure of the other, and so makes for less brittle deployments. Often times, the different domains are those of different companies, i.e. an enterprise and a SaaS provider, or two business partners but federation technology may also be deployed solely within a single company (quite often after an acquisition), and can provide the same advantages.
- **Level of Assurance (LOA)** – LOA is defined in ISO ISO/IEF 291165. A LOA typically represents, at minimum, both the user's authentication and the identity proofing and registration procedure by which the user's account was created. Various authentication methods provide different levels of confidence that the user is who they've claimed to be, due to qualities and limitations inherent to the methods. It is therefore useful for enterprises to adopt classifications of the different types of authentication so that those classes of authentication that represent different amounts of confidence can be leveraged in policy definition and enforcement. Agencies within the US Federal government have historically taken this approach – defining what are called "Levels of Assurance (LOA)" and the types of authentication that provide each level, as well as the types of applications that should be protected with each level.
- **Authenticator Assurance Level (AAL):** the most recent version of the NIST Digital Identity Guidelines broke apart the traditional LOA into IAL (Identity Assurance Level), AAL (Authentication Assurance Level), and FAL (Federation Assurance Level). AAL refers to the authentication process and is most

relevant to FIDO-based authentication. NIST Special Publication 800-63B defines three AALs expressing the level of confidence that the claim controls authenticator (s) bound to a subscriber's account.

3 Actors

There are some concepts that are common across federation protocols. In some cases these concepts/roles have different but related meanings. The goal of the following section is to describe them and how they relate to each other.

- **Authentication Authority (AA)** – a generic term for the role of the security domain that is responsible for authenticating a user and asserting that authentication to Application Providers in other security domains. Examples of Authentication Authorities are the Identity Provider in SAML and the OpenID Provider in OpenID Connect.
- **Authorization Server (AS)** – this is the term used in OAuth specification to describe the role of the actor that authenticates the user, collecting their consent, and issuing access tokens.
- **Application Provider (AP)** – a generic term for the role of the security domain that provides applications to end users, leveraging an authentication performed by an Authentication Authority. Examples of Application Providers are the roles of the Service Provider in SAML, and the Relying Party in OpenID Connect.
- **Service Provider (SP)** – this is the term used within the SAML specification to describe the role of the organization providing an application to an end user. In SAML, the service provider relies on an Identity Provider to authenticate users and provide information about them.
- **Identity Provider (IDP)** – the term that is used by the SAML specification to describe the role of the organization that authenticates an end user and provides information about that authentication and other user details to Service Providers (SPs).
- **Relying Party (RP)** – this term is used by both the FIDO and OpenID Connect specifications to define actors that play similar roles – that of the application provider that relies on another actor to authenticate the user. The FIDO specification defines the Relying Party as the application that is leveraging the authentication client to authenticate the user. The OpenID Connect specification uses 'Relying Party' to describe the role of the organization that provides an application or service to the end user.
- **OpenID Provider (OP)** – the role defined by the OpenID Connect (OIDC) specification for the entity that is responsible for authenticating an end user, and provides information about users to Relying Parties (RPs).

The following table details the names the different specifications use for specific roles:

	Authentication Authority	**Application Provider**
SAML 2.0	Identity Provider	Service Provider
OpenID Connect	OpenID Provider	Relying Party

4 Key Federation Standards

This section provides an overview of relevant federation standards for which integration with FIDO may be valuable.

4.1 SAML 2.0

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security and identity information between online business partners. It was developed by the Security Services Technical

Committee (SSTC) of the standards organization OASIS (the Organization for the Advancement of Structured Information Standards). This security information is expressed in the form of portable SAML assertions that applications working across security domain boundaries can trust. The OASIS SAML standard defines precise syntax and rules for requesting, creating, communicating, and using these SAML assertions.

4.2 OpenID Connect 1.0

OpenID Connect 1.0 is a profile & extension of OAuth 2.0, adding constructs in support of identity-based use cases like SSO and attribute sharing. OpenID Connect is defined within the OpenID Foundation (OIDF). Connect logically *layers* identity onto the OAuth base — stipulating how to use OAuth to 'do' identity, as opposed to other non-identity centric applications that are possible with OAuth.

4.3 OAuth 2.0

OAuth 2.0 is an IETF (Internet Engineering Task Force) standardized framework for API authentication and authorization. OAuth defines how a Client of an API can obtain security tokens that express a set of permissions against that API. These tokens are attached by the Client to its calls to the API and thereby serve to indicate the Client's authorizations with respect to the request it is making.

Though typically not deployed across policy domains, OAuth 2.0 may still be deployed with a FIDO-based authentication of the user into the Authorization Server.

5 Representing Authentication

A federation authentication authority and an application provider can refer to authentication events within protocol messages to varying degrees of specificity and granularity. Generally, the more specific, the greater a burden on the application provider for defining and maintaining policies on how to differentiate authentication types. The less specific, the simpler the burden for the Application Provider, but the more complicated for the Authentication Authority as it takes on the responsibility of mapping between the actual authentication event and whatever abstract representation is made to the application provider.

A number of organizations have come up with lists of Authentication Methods and Assurance Levels, and these SHOULD be used where possible instead of defining new ones. Over time new authentication methods emerge, and so these lists will not be exhaustive and so federations will need to agree on what to call those new methods when referring to them in protocol interactions.

We list different approaches below

1. Specify the actual authentication mechanism, e.g. 'password' or, in the FIDO context, a particular named FIDO authenticator.
2. Specify a policy describing the security characteristics of the authentication, e.g. 'phishing resistant'
3. Specify the ISO "Level Of Assurance", e.g. 'LOA1'. LOA provide a shorthand to refer to the multiple processes and procedures that, taken together, impact the degree of confidence or assurance the Authentication Authority has in its assertion of the user's identity.
4. Specify the Authenticator Assurance Level (AAL) as per NIST Special Publication 800-63B publication. NIST 800-63 distinguishes between three key (mostly orthogonal) security characteristics of the authentication event – these being

    - How strongly the person was identity proofed, which ties to their physical identity
    - How resistant a given credential is to attacks like impersonation, guessing, and theft
    - How strongly a given transaction's assertion is protected as it's passed over the network

In the FIDO context, it would be the AAL that would represent the FIDO-based authentication.

The above models vary in how the burden of understanding the impact of different authentication mechanisms and other processes on the overall assurance that can be ascribed to the user. Specifically, they vary in what the Application Provider must understand and have policy for. If the authentication is FIDO-based, the first approach above implies that the Application Provider would be sensitive to FIDO, the second and third imply that the Application Provider would be insulated from FIDO details, the Authentication Authority taking on that responsibility on the Application Provider's behalf.

When communicating that a particular FIDO Authenticator was used for the authentication, the Authentication Authority may make available to the Application Provider the details of that authenticator by providing a link to its metadata from services such as the FIDO Metadata Service (MDS).

Other business considerations may impact the level of information the Authentication Authority passes to the AP. For instance, the Authentication Authority may not wish to disclose the False Accept Rate (FAR) of the authenticator to the Application Provider due to the potential security and business sensitivity of this detail.

Similarly, the nature and closeness of the business relationship between Application Provider and Authentication Authority may impact how specific the Application Provider is in guiding the Authentication Authority's choice of authentication. For instance, if Application Provider and Authentication Authority are both within the same policy domain, the Application Provider could specify the nature of the transaction (e.g. money transfer) the user was attempting to perform, and influence the Authentication Authority in that way.

6 Representative Scenarios

The following scenarios are representative of how federation and FIDO-based authentication may be deployed in combination.

6.1 Web Single Sign-on

Federation protocols are primarily used to provide the application user with the benefit of having authentication events that occur be leveraged across application providers. There are two primary scenarios where a user performs authentication – when they initially log in, and when they attempt to access an application resource that requires additional/stronger authentication methods to be performed.

6.1.1 Initial login

Initial login is the scenario where a user first logs in to a service. In this scenario, federation protocols are used to enable an Application Provider to leverage a third-party Authentication Authority to authenticate its users. In the FIDO context, this request from the Application Provider to the Authentication Authority would stipulate that the authentication either be FIDO-based or otherwise guide the Authentication Authority to choosing a FIDO-based authentication from possible options. In its authentication response back to the AP, the Authentication Authority would stipulate that a FIDO-based authentication had occurred, or otherwise indicate to the Application Provider that the authentication has certain security characteristics or assurance.

6.1.2 Step-up Authentication

Step-up refers to a scenario where a user, having authenticated at the Authentication Authority with a particular authentication mechanism and having then been redirected to the Application Provider with an assertion to that fact, that assertion is subsequently determined by the Application Provider to engender insufficient confidence for a particular resource request. Consequently, the Application Provider redirects the user back to the Authentication Authority with a request they be re-authenticated with a different authentication mechanism.

This model is being normalized under the OASIS Trust Elevation Framework. In the FIDO context, the step-up request from the Application Provider to the Authentication Authority could either indicate that a FIDO-based authentication is desired to supplement the prior non FIDO-based authentication, or instead that a higher assurance FIDO-based authentication is desired to supplement a previous lower assurance FIDO-based authentication.

6.2 Delegated Authorization

If using OAuth, client applications authenticate to protected-APIs by presenting access tokens on their API calls. When those APIs protect user resources (e.g. calendar, step data, profile, etc.) the relevant user is typically asked for their consent to that access – which necessarily implies that the user be authenticated. Depending on the sensitivity of the resources accessed over the API, it could be relevant for the particulars of the authentication event to be represented in the OAuth access tokens issued to the client for use on the API calls.

6.2.1 Native application clients

For native application clients of APIs, one model is that the authentication of that user occurs in a browser window (either a system browser or preferably a browser tab) and not directly within the application's own user interface. Consequently, it would be in this browser window/tab that the FIDO authentication would occur. Once successfully authenticated via FIDO, the user may be presented with a consent page in which they can qualify the permissions the native application should be granted with respect to the API access. Subsequently, OAuth tokens are issued to the native application, for it to store locally and use on subsequent API calls.

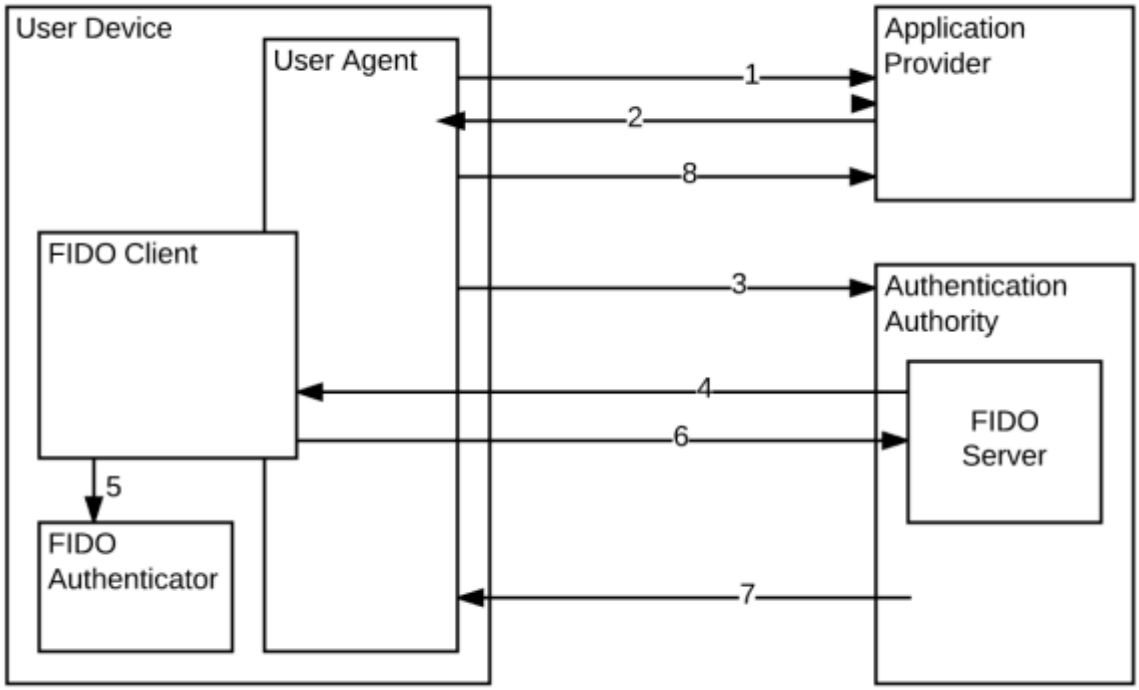
This model for authenticating users into native applications is being normalized and enabled by the OAuth for Native Applications Best Current Practice specification and the corresponding AppAuth SDK, which was developed by the OpenID Foundation.

6.2.2 Web server clients

For web server clients of APIs, the user will typically be required to authenticate when granting their consent for a web server client to access their data behind some cloud API. This consent and the relevant identities are captured (either implicitly or explicitly) in OAuth 2.0 tokens issued to that client. Because this consent step is critical to the underlying privacy and security model, the stronger the authentication of the user as part of this process the better. Consequently, authenticating the user via FIDO can be preferred. This preference for FIDO-based authentication could be simply a policy of the OAuth Authorization Server (i.e. use FIDO when possible) or could be driven by the OAuth Resource Server policies.

7 Integration Model

The high-level integration between a federation protocol flow and a FIDO-based authentication is shown below. This call flow assumes that the User registered FIDO credential with the Authentication Authority.



Combined FIDO & Federation High-level Flow

1. User accesses Application Provider (via User Agent) and needs to authenticate.
2. **Application Provider redirects the User Agent to the FIDO-enabled Authentication Authority with a request that the user be authenticated. This redirect allows the Application Provider to specify that the user authentication be FIDO-based (either explicitly or implicitly) according to its preferences.**
3. User Agent accesses the Authentication Authority federation endpoint. The Authentication Authority determines that the FIDO authentication specified in the previous request is both relevant and possible by confirming that the policies of the Authentication Authority match with the preferences of the Application Provider.
4. The Authentication Authority sends a FIDO Server challenge to FIDO Client (via User Agent). This challenge may indicate a particular FIDO Authenticator – as determined by the Application Provider's preference and the Authentication Authority's own policies.
5. FIDO Client locates FIDO Authenticator, and the user is authenticated
6. FIDO Client returns FIDO authentication response (via User Agent)
7. FIDO Server validates the FIDO response and reports same to Authentication Authority which looks up the user **and then redirects the User Agent back to the Application Provider with an authentication assertion**.
8. User Agent calls Application Provider with the authentication assertion (or an artifact that the Application Provider can exchange against the Authentication Authority for the actual authentication assertion, step not shown). The authentication assertion can include the specifics of the FIDO-based authentication or more generally a policy identifier for that authentication – as previously discussed.

The bolded steps are those by which respectively,

a) the Application Provider is able to indicate to the Authentication Authority its expectations for how the user *should* be authenticated and
b) the Authentication Authority is able to indicate to the Application Provider details about how the user *was* actually authenticated.

8 Recommendations

Regardless of how authentication policies are represented, it is recommended that Application Providers and Authentication Authorities use the appropriate message parameters that the federation protocols provide, as discussed below

8.1 SAML 2.0

SAML provides a number of parameters in support of authentication specifics.

8.1.1 AuthnRequest

When initiating an SSO flow, the Application Provider can optionally send specifics about the way it would like the authentication performed by specifying the corresponding URI in the <RequestedAuthNContext> element.

Example:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="809707f0030a5d00620c9d9df97f627afe9dcc24"
Version="2.0" ProviderName="SP test" IssueInstant="2014-07-16T23:52:45Z"
Destination="http://idp.example.com/SSOService.php"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
AssertionConsumerServiceURL="http://sp.example.com/demo1/index.php?acs">

…
  <samlp:RequestedAuthnContext Comparison="exact">


<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</saml:AuthnContextClassRef>
  </samlp:RequestedAuthnContext>
…
</samlp:AuthnRequest>
```

8.1.2 Response

A SAML response typically contains a SAML assertion that contains an <AuthnStatement> element that the Authentication Authority uses to convey information about the authentication event performed by the user. Part of the <AuthnStatement> element is the <AuthnContext> which is designed to provide such details about the authentication event.

Example:

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_8e8dc5f69a98cc4c1ff3427e5ce34606fd672f91e6"
Version="2.0" IssueInstant="2014-07-17T01:01:48Z"
```

```
Destination="http://sp.example.com/demo1/index.php?acs"
InResponseTo="4fee3b046395c4e751011e97f8900b5273d56685">
…
  <saml:AuthnStatement AuthnInstant="2014-07-17T01:01:48Z" SessionNotOnOrAfter="2024-07-
17T09:01:48Z" SessionIndex="_be9967abd904ddcae3c0eb4189adbe3f71e327cf93">
    <saml:AuthnContext>

<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</saml:AuthnContextClassRe
f>
</saml:AuthnContext>
</saml:AuthnStatement>
…
</samlp:Response>
```

8.2 OpenID Connect 1.0

OpenID Connect provides mechanisms modeled after those of SAML by which Application Providers and Authentication Authorities can communicate information about authentication events.

On the Authentication Request, the Application Provider can stipulate its policy requirements of the authentication using the 'acr_values' parameter. In its response, the Authentication Authority will report back the policy under which the authentication was performed using the 'acr' claim, and optionally specify the particular authentication mechanism(s) using the amr' claim.

For instance, the Application Provider can request an authentication that meets policy 'Silver' – which has been previously defined to require multi-factor with in-person identity proofing. That the Silver policy was met by the authentication (and prior proofing) is reported back in the 'acr' and the Authentication Authority can optionally provide more info in the 'amr' list to indicate that the user used a password and hardware backed U2F token for that particular session.

The OpenID Connect specification does not itself define any particular values for either the 'acr' or 'amr'.

The IETF AMR specification is defining an IANA registry for AMR policy identifiers. The specification will also define a limited set of authentication mechanism identifiers – such as 'face', 'fpt', 'geo', 'hwk', 'iris', 'kba', 'mca', 'mfa', 'otp', 'pin', and 'pwd'.

Separately, the OIDF has released the OpenID Connect Extended Authentication Profile (EAP) ACR Values specification that defines particular 'acr' policy identifiers, namely

- phishing-resistant
- phishing-resistant-hardware

8.2.1 Authentication Request

In order to direct the Authentication Authority to perform a FIDO authentication, the Application Provider SHOULD specify a value for the 'acr_values' in its Authentication Request. It is recommended that the 'acr_values' use either of the values defined by the OIDF EAP profile.

8.2.2 Response

After a successful FIDO Authentication, the Authentication Authority SHOULD specify the appropriate authentication policy identifier in the 'acr' claim in the 'id_token'. It is recommended that the 'acr' claim use the values defined by the OIDF EAP profile.

Additionally, the Authentication Authority MAY specify the particulars of the FIDO authentication by adding appropriate authentication identifiers to the 'amr' claim in the 'id_token'. It is recommended that the 'amr' claim use values from the list defined by the IETF AMR specification.

8.3 OAuth 2.0

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to a service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the Authorization Server, or by allowing the third-party application to obtain access on its own behalf.

In a common pattern for OAuth, the resource owner authenticates to the Authorization Server before the Authorization Server issues an access token to the client application that is used on API calls to the Resource Server. The Authorization Server can choose to use any authentication mechanism, including FIDO. As the authentication often precedes the user giving their consent to the client being given API access so the user's resources at the RS, the particulars of the authentication could well matter to the RS. In support of this, the Authorization Server could insert a representation of the authentication event into the access token it issues.

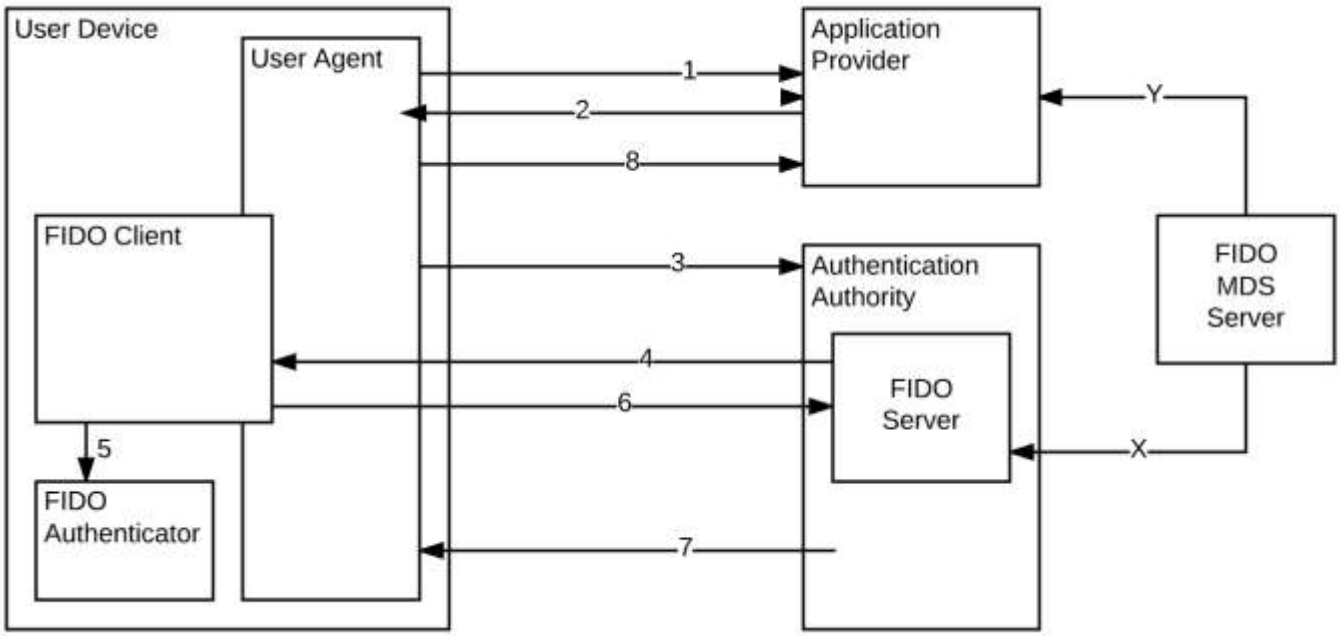
9. Appendices

8.4 Extended model using MDS

FIDO defines a standard means to describe the relevant pieces of information about an authenticator in order to interoperate with it, or to make risk-based policy decisions about transactions involving a particular authenticator. Such information is called Metadata and FIDO standardizes Metadata Statements. FIDO provides an MDS (Metadata Service) where Metadata of various FIDO authenticators can be freely downloaded and used by anyone.

As for federation protocols, Metadata are useful for the Authentication Authority and Application Provider to characterize the security levels of the authentication in detail. For example, Metadata are useful for the Authentication Authority to abstract the levels of the user authentication. In addition, when the Application Provider requires verification of risks scored by authentication context of Authentication Authority MDS can provide Metadata of authenticators to help enable the verification.

The following model (option) is shown to supplement the content in Section 7.



Metadata from MDS are typically downloaded frequently and cached in local storages of Authentication Authority (X) and Application Provider (Y) in the above figure.

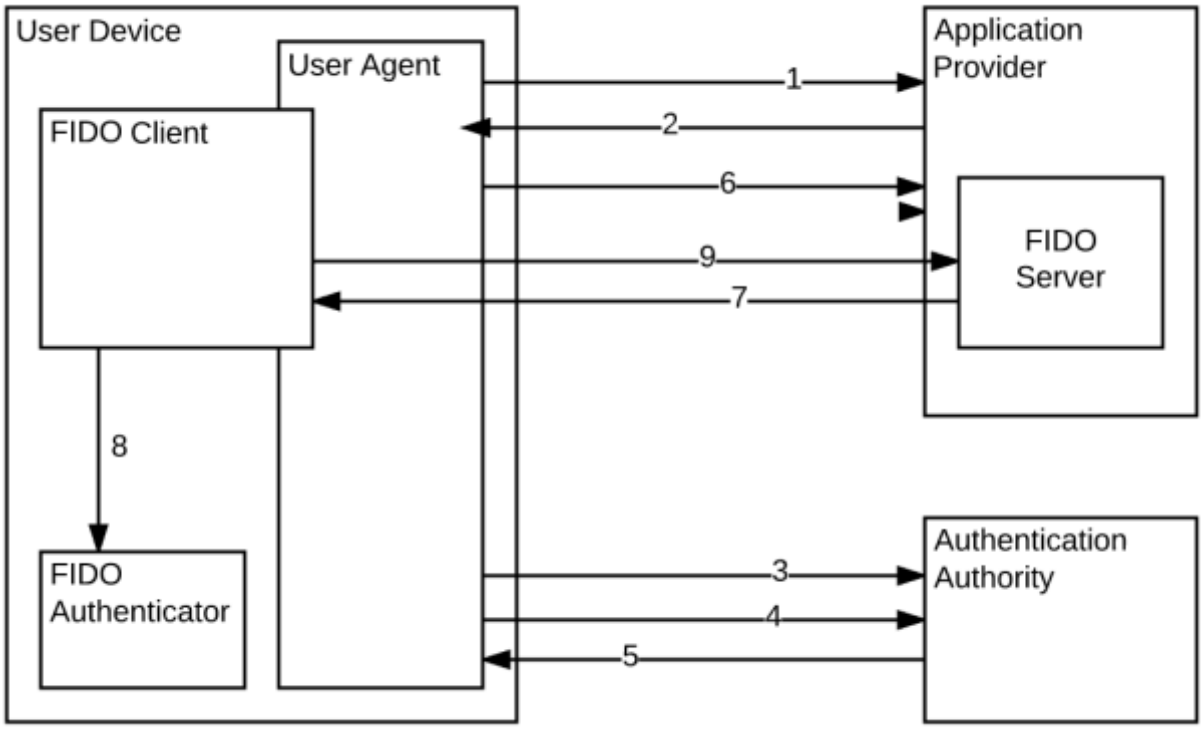
For Authentication Authority subscribing to MDS is highly recommended in order to perform authentication from FIDO authenticators. It is also recommended for APs to do the same if the Application Provider needs detailed information subject to their business model. Since both parties can access the same information from MDS, Authentication Authority is required to send only the identifier of the authenticator, e.g., AAID or AAGUID, to the Application Provider.

On the other hand, if the Authentication Authority and the Application Provider want to deploy more controlled models, the Authentication Authority may use Metadata from the MDS and create abstracted information for the Application Provider. In such a case, the Authentication Authority may not disclose the identifier of the authenticator to the Application Provider.

8.5 FIDO-Enabled Application Provider

FIDO-enablement of Authentication Authorities is a simple means of ensuring the benefits of FIDO authentication can be realized across multiple applications in an enterprise deployment. However, enterprises with existing identity federation solutions may not upgrade their identity and access management systems until FIDO reaches a certain level of penetration amongst their subscribers. This should not hinder the use of FIDO in an identity federation context. In such a scenario, a given enterprise could act as both FIDO Relying Party & Federation Application Provider.

One potential model for use of FIDO for such FIDO-enabled Application Providers is shown below:



With respect to the above diagram:

1. User accesses Application Provider.
2. Application Provider directs User Agent to log into Authentication Authority
3. User Agent accesses Authentication Authority
4. User authenticates to Authentication Authority with non-FIDO-based authentication mechanism.
5. Authentication Authority provides authentication confirmation (e.g. authentication token) via redirect URI to AP.
6. User Agent accesses redirect URI - providing Authentication Authority authentication confirmation (e.g. token). Application Provider server validates Authentication Authority authentication token. Application Provider determines that additional assurance is required
7. RP sends FIDO Server challenge to FIDO Client to prompt authentication.
8. FIDO Client prompts FIDO Authenticator to get assertion.
9. FIDO Client sends assertion to RP.

The flow provides a way for an FIDO-enabled RP to leverage a non-FIDO-enabled Authentication Authority In this example, the Application Provider creates a FIDO credential as part of registration, but also directs the user to log into an Authentication Authority. When the Authentication Authority verifies the user identity, it sends the assertion to the AP. In this way, the Application Provider can now associate the FIDO credential with the Authentication Authority assertion.

Editors

Paul Madsen, Ping Identity

Darren Platt, RSA

Salah Machani, RSA

Acknowledgements

The editors would like to thank all FIDO members who reviewed or contributed to this paper, namely Giridhar Mandyam, Max Hata, Wataru Ogami, Avinash Umap, Anthony Nadalin, and Brian Wood.