BuildAFederation

How to Build a Federation

Shibboleth federations provide a scalable infrastructure to help providers discover each other, identify each other, and understand the information they share. They are not necessary to run Shibboleth in a production environment, but they can simplify things greatly. This document provides some advice on how to move from a skeletal deployment to a full-fledged federation.

This guide can be used to make an intra-campus SSO system, to provide federated access amongst a community, or both. Before you start, check that there is no pre-existing federation that meets your needs. If there is, it would be simpler to join that federation and leverage the work they've already done.

1. Create the Metadata

Metadata provides the basis for all trust between providers in Shibboleth. The metadata doesn't configure the provider itself, but is used instead to identify and describe counterparts. When an SP or an IdP receives a request from another provider, it needs to be able to verify that the remote provider is who it claims to be. This is done by comparing the declared name in the SAML message to the names the provider knows from the metadata. When there's a match, the remote provider must then present the credentials that are in that provider's metadata. Now that the remote provider has been positively identified, they can communicate. If there is no match, or the credentials presented are wrong, no attributes will be sent.

The metadata file is either signed, hosted using https://, or both, and frequently downloaded by all providers in the federation. You can either aggregate self-generated metadata from all the providers by accessing their entityID (or Metadata handler), or write the metadata file by hand following this example.

Entity Naming

The https://yourFederationName/federation/metadata.xml value should be replaced with one appropriate for your federation. Make sure to use a legal namespace you control. This <EntitiesDescriptor> element is filled with one or more <EntityDescriptor> elements, uniquely named, with one for each provider.

Standard practices use https://your-hostname/idp/shibboleth and https://your-hostname/shibboleth-sp, but these are only conventions. We think it's best if the entityID can be resolved to the metadata handler. You can use any URI you want. These names must match the values of the entityID specified in shibboleth2.xml and relying-party.xml.

Trust

Shibboleth relies on public-key cryptography certificates to allow providers to identify one another through both signed assertions and SSL/TLS. The certificates used by your providers in these transactions must match the corresponding data in their metadata entries. This can be achieved by either using key names and PKI or by placing the full certificate data directly in the metadata.

Placing the certificate directly in the metadata is strongly recommended for several reasons. Future versions of Shibboleth will allow assertions to be encrypted and pushed directly to providers, simplifying flows and enabling new deployment scenarios. This is only possible if the public key of the relying party is available. Many of the hassles that accompany PKI deployment, such as revocation, can be handled using the metadata more easily – simply replace the certificate with a new one. Additionally, any signer can be used as long as the private key is sufficiently well-guarded. This does not preclude the use of a common root.

If the machine running a provider already has a certificate, it can be used as long as it's being embedded or its root is in the metadata file. If you need to generate certificates, make sure to generate the public/private keypair securely, such as on the machine itself. Centrally generating keys is insecure and only used in TestShib for convenience. Once the keys are generated, either self-sign them on the machine, or sign them with one of the trusted roots included in the metadata.

Shibboleth allows distinct certificates to be used when the provider is operating in different capacities, such as one for the SSO service and another for the AA. This will rarely be encountered, and the same certificate should be present in both parts of the metadata.

Scope

Identity Providers have an extra extension in their metadata to handle attribute scope. This extension limits the attribute scopes that a provider is allowed to be authoritative for; for example, Supervillain University couldn't send an eduPersonScopedAffiliation of member@brown.edu (unless the federation decided it wanted to allow Supervillain University to express that someone was a member of Brown). This has important policy implications since the scope of an attribute is often used for trust decisions. For Supervillain University’s IdP, this would only be <shibmd:Scope>supervillain.edu</shibmd:Scope>.

Signing

While hosting the metadata behind SSL/TLS provides some degree of integrity and authentication, metadata may additionally be signed by the central authority to ensure it hasn't been tampered with. Providers can use a signature filter on the metadata file they load to check the signature against a certificate of your choice.

2. The Discovery Problem & WAYF

One of the hardest problems in federated identity is the determination of the Identity Provider for a user. The essential goal is to redirect the user's browser to the location of the SSO endpoint with information about this request encoded in the URL. While Cardspace offers a long-term solution for the problem, the most common approach now is to use metadata to drive a user interface. This interface may be operated by the SP, often powered by metadata the federation maintains, or as a DS operated by a third party or the federation itself.
If an application will generally only allow users from a single IdP to access it, its SessionInitiator should point directly at the SSO service, sidestepping discovery entirely. The default Login example uses that.

The traditional approach is for the federation to stand up a central DS service through which users at all service providers are funneled.

3. Configure the Providers

We assume you already have operational providers installed and tested against a service such as TestShib Two.

The configuration of all your providers needs to match the information contained in the metadata. Configuration of the providers is covered in-depth in the Configuration section of the Wiki. The only other change required is the addition of a `<MetadataProvider>` that points to the URL where you're hosting the metadata file.

4. Create the Policy

Every federation should establish and publish a document that describes its policies, practices, and operation. There is a lot of accumulated experience in federation development by this point, and we know roughly what the standard principals and points of divergence are. It's best to refer to existing federation policy documents as a foundation and consider the own unique needs and legal restrictions of your jurisdiction and community.

5. Create the Attributes

The final part of an exchange between providers that may be facilitated by a federation is the definition of attributes that would be of common interest to federation members. This provides a common language for expressing and interpreting data about users so that access is properly controlled.

Attribute names in SAML, and by extension Shibboleth, are generally expressed as URIs. Section 8.2 of SAML 2.0 Profiles recommends that all LDAP attributes be expressed as URIs through transformation of the OID identifier, e.g. `urn:oid:1.3.6.1.4.1.5923.1.1.1.5`. Attributes may also be named through other URN namespaces, or by using URLs, which could be resolved to yield a controlled vocabulary, synonyms, and other information. It's very important that proper attribute naming etiquette be followed and that you don't define attributes in namespaces you don't control.

Proliferation in attributes and misappropriation of attributes are two countervailing forces that need to be balanced as you decide which attributes to use. A large number of attributes has already been defined and named (e.g. `eduPerson` and `x.520`), and in any situation where a pre-existing attribute meets your needs, it should be used. However, controlled vocabularies and specific definitions must be followed. Use your best judgment in deciding whether to create a new attribute definition.

Attribute definitions should be written in appropriately normative language and assigned at least a SAML-based name. This information should be made available to all providers in your federation and referenced by or included in your federation's policy statement.