IdPNameIdentifier

Supporting a new Name Identifier

Before continuing you should understand the concept of a name identifier and how to define and release attributes.

Supporting a new name identifier within the identity provider includes:

1. Configuring the IdP to produce the name identifier
2. Configuring the IdP to accept the name identifier (optional)

The second step is occasionally important, but not necessarily mandatory, depending on the requirements of the relying parties you need to support. Specifically, the ability to reverse the identifier back into a user's identity is essential for supporting back-channel queries, among other features, but is not strictly needed for a one-way communication path such as is used by default with SAML 2.0 SPs. Most of the time, if you're just doing this to accomodate a vendor with a lousy SAML implementation, you can ignore that step.

Producing the Name Identifier

As discussed, name identifiers have different sets of properties (e.g. longevity, transparency). Therefore it should come as no surprise that there are multiple ways in which to produce the name identifier depending on which properties are required.

The following name identifier types are supported by the IdP out of the box.

Transient	An identifier that is transient and opaque
Persistent	An identifier that is persistent and opaque

Pursuant to the note above, if you're prepared to forgo the IdP's ability to reverse the identifier back into a user identity, you can generate an arbitrary identifier of any format. Alternatively, you could develop your own custom `PrincipalConnector` to perform the reverse mapping process.

Custom	An arbitrary identifier format

Accepting the Name Identifier

The second part of the configuration allows the IdP to map a name identifier back in to an identifier for the user (e.g. their login id). This identifier is known as a principal name and so the plugin that connects the name identifier to a principal name is known as the `PrincipalConnector`.

The following `PrincipalConnector` types are available out of the box.

Transient	maps from an identifier created by the TransientId attribute definition
CryptoTransient	(2.3+) maps from an identifier created by the CryptoTransientId attribute definition
Stored Id	maps from an identifier created by the StoredId data connector
Direct	assumes the value of the name identifier is the user's principal name and as such performs no mapping

Expressing Support in Metadata

An IdP can express support for a name identifier format through its metadata. This is done by adding a `<NameIDFormat>` element to **both** the IdP's `IDPSSODescriptor` and `AttributeAuthorityDescriptor` roles. The value of this element should be the format as configured in the name identifier attribute encoder.

Example NameIDFormat expressing support for persistent identifiers

```
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
```

In practice, little if any software uses this information, so other than being useful as a documentation aid, it's not usually important to do this.

IdP Name Identifier Selection Process

An identity provider is often configured to produce multiple types of name identifiers. The process by which the IdP selects the name identifier format to use for a given relying party involves two steps.

First, all the attributes resolved for the user are winnowed down to a set that might potentially be encoded as a name identifier for this request by performing the following filtering steps:

1. The attributes are filtered by the protocol used to contact the IdP. If SAML 1 was used, then only attributes that may be encoded as a SAML 1 `NameIdentifier` are retained; likewise for SAML 2.
2. The remaining attributes are filtered by the name identifier formats listed as supported in the SP's metadata, if any. Note that the IdP treats the "unspecified" format defined by SAML as a wildcard and if it appears in an SP's metadata, the IdP will assume that the SP can accept any format it produces.

If no attributes remain after the filtering step, no name identifier is generated. Otherwise, the IdP moves on to the second stage: selection of a single attribute from the set of possible attributes. It does so using the following steps:

1. If the service provider requires a particular format, the attribute that encodes to that format is selected. If no attribute encodes to that format, an error is returned to the SP.
2. If there is a name identifier format precedence list for the relying party, the IdP selects the attribute that encodes to the format with the highest precedence.
3. If there is no name identifier format precedence list, or no attribute encodes to a format listed in the precedence list, then the IdP chooses an attribute at random.