NameIDGenerationConfiguration

Current File(s): *conf/saml-nameid.xml, conf/saml-nameid.properties*

Format: Native Spring / Deprecated Custom Schema

Legacy V2 File(s): *conf/attribute-resolver.xml*

Overview

Generation of SAML NameIdentifier/NameID content is handled by the NameIdentifierGeneration service. See the NameIdentifiers topic for a general discussion of name identifiers and a list of specific examples.

The *saml-nameid.xml* file is used to control the generation of SAML 1 NameIdentifier and SAML 2 NameID content. SAML assertion subjects contain a special slot for an identifier that is less commonly used in Shibboleth deployments (because SAML Attributes are more general and useful) but is very commonly used by vendors seeking to do the bare minimum necessary to support SAML.

When interoperating with Shibboleth SPs, it's rare to need to modify this file, but you might need to do so to add support for more application-oriented identifier types, such as email addresses, or less commonly to enable support for so-called "persistent" identifiers, special privacy-preserving identifiers that are targeted to specific services.

Generators

The configuration defines two list beans, each containing a list of "generator" plugins for the two different SAML versions. Each plugin is specific to an identifier Format, a SAML constant that identifies the kind of value being expressed. The generation process involves selecting a list of Formats to try and generate (see Format Selection below), and then trying each Format until an appropriate value is obtained by running each configured generator in order. Since assertions need not contain a name identifier, it is not an error (from the perspective of the IdP) for all the generators to fail.

The default configuration includes generators for "transient" identifiers. These plugins are configured using *saml-nameid.properties* to control the strategies used to generate and reverse-map the values (the latter only being necessary to support "back-channel" attribute queries).

In the case of SAML 2, a plugin is present, but commented out, to generate "persistent" identifiers. Certain properties in *saml-nameid.properties* must be set in order to safely uncomment this plugin (discussed below).

The default configuration also demonstrates how to generate a custom identifier using an arbitrary Format based on an attribute from the attribute resolution process. This mirrors the V2 approach of encoding attributes, but separates the two operations in the configuration. This plugin also has the capability, unlike the resolver-based approach, of selecting the first value present from a list of possible source attributes.

In summary:

- Support for "transient" identifiers is automatic.
- If you want "persistent" / pair-wise support, see below.
- If you want custom values, see below.

Format Selection

For any given request, the ordered list of Formats to try to generate is based on combining the SP's request (SAML 2 requests can include a `<NameIDPolicy>` element that requires a particular Format), the `<NameIDFormat>` element(s) in the SP's metadata, and the `nameIDFormatPrecedence` profile configuration property, if set for the chosen relying party configuration. If the metadata contains nothing, or contains the "`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`" value, then the metadata is ignored.

If a `<NameIDPolicy>` element with Format is supplied, a suitable identifier MUST be generated or an error will be returned.

Otherwise the formats specified in an SP's metadata are filtered against a `nameIDFormatPrecedence` profile configuration property, if set, and the resulting set of Formats is tried in order. That is, the first Format in the profile configuration that is also in the metadata and that results in a valid result will be used.

If metadata is specified but no profile setting is, then the metadata is used as is, and vice versa.

Default Formats for each SAML version are set via *saml-nameid.properties* and are used in the event that nothing else is called for. You should **not** alter that setting in most cases.

Transient Identifier Generation

The strategy used to generate transient identifiers is controlled with the **idp.transientId.generator** property in *saml-nameid.properties*.

The default strategy is based on the use of a secret key, discussed in the SecurityConfiguration topic (see the idp.sealer.* properties). This maintains support for attribute queries without requiring shared state between a cluster of nodes, apart from sharing the secret key.

You can set this property to "shibboleth.StoredTransientIdGenerator" to generate random values tracked by server-side storage (this makes them shorter, but requires more complex storage approaches when clustering if attribute queries need to be supported).

Persistent Identifier Generation

See the PersistentNameIDGenerationConfiguration subtopic for detailed help with this feature.

Custom Identifier Generation

See the CustomNameIDGenerationConfiguration subtopic for detailed help with this feature.

More commonly in the case of custom formats, if you also need to adjust other behavior such as the content of the `NameQualifier` or `SPNameQualifier` attributes, refer to the javadocs for the generator interfaces (see reference table below for links).

Reference

Beans

Beans defined in *saml-nameid.xml* and related system configuration follow:

Bean ID	Type	Function
shibboleth. SAML2NameIDGenerators	List<SAML2NameIDGenerator>	SAML 2 NameID generator plugins to use
shibboleth. SAML1NameIdentifierGenerators	List<SAML1NameIdentifierGenerator>	SAML 1 NameIdentifier generator plugins to use
shibboleth. SAML2TransientGenerator shibboleth. SAML1TransientGenerator	SAML2NameIDGenerator SAML1NameIdentifierGenerator	Plugins for generating transient identifiers using pluggable strategies
shibboleth. StoredTransientIdGenerator	TransientIdGenerationStrategy	Strategy plugin that generates transient identifiers randomly and stores them in a server-side StorageService
shibboleth. CryptoTransientIdGenerator	TransientIdGenerationStrategy	Strategy plugin that generates transient identifiers by encrypting a subject identity into a long opaque string
shibboleth. SAML2PersistentGenerator	SAML2NameIDGenerator	Plugin for generating persistent identifiers using pluggable strategy
shibboleth. ComputedPersistentIdGenerator	PersistentIdGenerationStrategy	Strategy plugin that generates persistent identifiers with a salted hash of an input value
shibboleth. StoredPersistentIdGenerator	PersistentIdGenerationStrategy	Strategy plugin that generates persistent identifiers and stores them in a database
shibboleth.JDBCPersistentIdStore [3.2]	JDBCPersistentIdStoreEx	Parent bean for defining a JDBC store for persistent identifiers
shibboleth. SAML2AttributeSourcedGenerator shibboleth. SAML1AttributeSourcedGenerator	SAML2NameIDGenerator SAML1NameIdentifierGenerator	Template beans for plugins that generate custom identifiers based on resolved and released attribute values
shibboleth. LegacySAML1NameIdentifierGenerator shibboleth. LegacySAML2NameIDGenerator	SAML2NameIDGenerator SAML1NameIdentifierGenerator	Plugins supporting deprecated use of attribute resolver configuration to produce and encode name identifiers

Properties

Properties defined in *saml-nameid.properties* to customize various aspects of default identifier generation behavior:

Property	Type	Default	Function

idp.transientId.generator	Bean ID of a TransientIdGenerationStrategy	shibboleth.CryptoTransientIdGenerator	Identifies the strategy plugin for generating transient IDs
idp.persistentId.generator	Bean ID of a PersistentIdGenerationStrategy	shibboleth.ComputedPersistentIdGenerator	Identifies the strategy plugin for generating persistent IDs
idp.persistentId.dataSource 3.2	Bean ID of a JDBC DataSource		Identifies a data source for storage-based strategy for persistent IDs
idp.persistentId.store	Bean ID of a PersistentIdStore		Identifies the data store plugin for storage-based strategy for persistent IDs
idp.persistentId.computed	Bean ID of a ComputedPersistentIdGenerationStrategy	shibboleth.ComputedPersistentIdGenerator	May be null, Identifies a strategy plugin to use to generate the first persistent identifier for each subject, used to migrate from the computed to stored strategies
idp.persistentId.sourceAttribute	Comma-delim'd List		List of attributes to search for a value to uniquely identify the subject of a persistent identifier, it MUST be stable, long-lived, and non-reassignable
idp.persistentId.useUnfilteredAttributes 3.2	Boolean	true	Whether or not the previous property has access to unreleased attributes
idp.persistentId.salt	String		A secret salt for the hash when using computed persistent IDs
idp.persistentId.encodedSalt 3.3	Base64-encoded String		An encoded form of the previous property
idp.persistentId.algorithm	String	SHA	The hash algorithm used when using computed persistent IDs
idp.persistentId.encoding 3.3.2	"BASE64" or "BASE32"	BASE64	The final encoding applied to the hash generated when using computed persistent IDs (BASE32 is strongly recommended for new installs)
idp.persistentId.exceptionMap 3.4	Bean ID	shibboleth.ComputedIdExceptionMap	Advanced feature allowing revocation or regeneration of computed persistent IDs for specific subjects or services
idp.nameid.saml2.legacyGenerator	Bean ID		**DEPRECATED** Identifies a default generator plugin to use as a last resort if no others succeed
idp.nameid.saml1.legacyGenerator	Bean ID		**DEPRECATED** Identifies a default generator plugin to use as a last resort if no others succeed
idp.nameid.saml2.default	URI	urn:oasis:names:tc:SAML:2.0:nameid-format:transient	The default Format to generate if nothing else is indicated
idp.nameid.saml1.default	URI	urn:mace:shibboleth:1.0:nameIdentifier	The default Format to generate if nothing else is indicated

V2 Compatibility

The V3 IdP uses a new dedicated service for configuring NameID generation. The legacy V2 approach of encoding attributes into identifiers using *attribute-resolver.xml* and special attribute encoders that generate NameIdentifiers or NameIDs instead of Attributes is supported for compatibility purposes, but is **deprecated** and may be removed from a future version.

To enable the legacy support, the **idp.nameid.saml2.legacyGenerator** and **idp.nameid.saml1.legacyGenerator** properties must be uncommented and set to the values commented out in the *saml-nameid.properties* file. This is done for you when performing an upgrade from V2.

The IdP should load any existing V2 *attribute-resolver.xml* file and configure itself in an expected manner, but that configuration will be superseded by the content of the new *saml-nameid.xml* file and will fall back to the resolver only as a backstop. You can short-circuit the new functionality by commenting out the content of the two generator list beans and leaving them empty.

Note that unlike in V2, the transient or persistent identifiers produced by the new V3 generation service are not treated as attributes and are not release-controlled via an attribute filter policy. Rather, transients are viewed as harmless (because they are merely one-time values) and persistent identifiers cannot be generated without configuring an appropriate source attribute or other properties.

Finally, note that the two supplied strategies for generating persistent IDs are compatible with the connector plugins in V2 used for this purpose.

Notes

TBD