ConsentConfiguration

Files: *conf/idp.properties, conf/intercept/consent-intercept-config.xml, messages/message.properties, views/intercept/attribute-release.vm, views/intercept/terms-of-use.vm*
Format: Native Spring

Overview

IdPv3 includes the ability to enforce user consent to attribute release as well as terms of use.

Consent to attribute release is enabled by default in new IdPv3 installations. It is not enabled during upgrades from IdPv2, but will become enabled if the legacy *conf/relying-party.xml* is replaced with the v3 default configuration.

Consent in IdPv3 was modeled after the uApprove and uApproveJP plugins to IdPv2.

It is not possible to upgrade consent data from uApprove to the IdPv3 implementation because the storage formats are not compatible, and there are no plans at present to provide such a migration path.

Consent is implemented by the `intercept/attribute-release` and `intercept/terms-of-use` intercept flows.

Attribute Release Consent

Attribute release consent requires users to accept the release of attributes to Service Providers during front-channel authentication profiles that include attribute data in the response.

Note the "front-channel" caveat above. The default configuration prevents the consent intercept from imposing itself if it detects that attributes would not be included in the response and would instead perhaps be accessed via a back-channel query. This is true by default with SAML 1.1, for example.

The system does not currently support the application of consent decisions by a user to the data released in a back-channel query. If you intend to use the consent feature, it is your responsibility to ensure that attributes would not be accessible to the same relying parties via query or some other means. The system will not prevent this from happening if you leave features enabled that would allow this to happen.

Users are prompted to consent to attribute release :

1. on initial access to service provider resources
2. on release of an attribute to which consent has not been previously given
3. when an attribute previously consented to is no longer released
4. (optionally) when the value of an attribute previously consented to changes, see Attribute Release Value Comparison.

Enabling Attribute Release Consent

Attribute release consent is enabled by default for the SAML 1 and SAML 2 SSO profiles via the `postAuthenticationFlows` property in *conf/relying-party.xml* :

Default relying party configuration

```
<bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
    <property name="profileConfigurations">
        <list>
            <bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" />
            <ref bean="SAML1.AttributeQuery" />
            <ref bean="SAML1.ArtifactResolution" />
            <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
            <ref bean="SAML2.ECP" />
            <ref bean="SAML2.Logout" />
            <ref bean="SAML2.AttributeQuery" />
            <ref bean="SAML2.ArtifactResolution" />
        </list>
    </property>
</bean>
```

Disabling Attribute Release Consent

To disable attribute release consent, remove the `attribute-release` post authentication flow from the profile configurations in *conf/relying-party.xml*.

For example, replace :

```
<bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" />
<bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
```

with :

```
<bean parent="Shibboleth.SSO" />
<bean parent="SAML2.SSO" />
```

Per Attribute Consent

To allow users to select the attributes they wish to be released, enable per attribute consent via `idp.consent.allowPerAttribute` in *conf/idp.properties*.

Attribute Release Consent Duration

Users may choose from three options when deciding the duration of their consent to attribute release. This functionality was derived from uApproveJP.

The duration options and descriptive text may be customized in *messages/messages.properties*.

Duration Option	Description

"Ask me again at next login"	Users will be prompted to consent to attribute release at every log in. This is implemented by not storing consent. **Note:** Consent is not activated when user attributes are retrieved by an SP via the back-channel (e.g. SAML Attribute Query). Even though a back-channel request technical is not a "login" by the user, some users might misunderstand the implications of choosing this option.
"Ask me again if information to be provided to this service changes"	The default. Users will be prompted to consent to attribute release if attributes have changed since consent was previously given. **Note:** Consent is not activated when user attributes are retrieved by an SP via the back-channel (e.g. SAML Attribute Query).
"Do not ask me again"	Optional. Users will not be prompted to consent to attribute release again. **All** attributes will be released to **any** service provider. This was called "global consent" in uApprove. The presence of this option is controlled by idp.consent.allowGlobal.

Attribute Release Value Comparison

By default, users are prompted to consent to attribute release if a "new" attribute is released or if an "old" attribute is no longer released. "New" and "old" in this context indicate whether consent has already been given to an attribute ID regardless of the attribute's values. In other words, by default, users are not prompted to consent to attribute release if an attribute's values change.

To prompt users to consent to attribute release if the values of an attribute have changed, set `idp.consent.compareValues` to `true` in *conf/idp.properties*.

Prompting users to consent to attribute release if an attribute's value changes requires additional storage capability, because the hash of an attribute's values must be stored for comparison. If client-side storage (cookies) are used to store consent, comparing attribute values may reduce the number of records that may be stored. Since a consent record is stored for every Service Provider, this may increase how often users are prompted to consent to attribute release.

Attribute Display

By default, users are prompted to consent to release all attributes except for the blacklisted attribute IDs `transientId`, `persistentId`, and `eduPersonTargetedID`. Blacklisted attributes are released to relying parties but are not displayed to users. A whitelist, blacklist, and match expression determine whether consent should be obtained for an attribute based on the attribute ID.

To prevent an attribute from being displayed, add the attribute ID to the blacklist. If a match expression is defined in addition to a blacklist, make sure that the blacklisted attribute ID does not match the match expression. The match expression overrides the blacklist, consequently a blacklisted attribute will be displayed if it matches the match expression.

Type	Description	Bean in *conf/intercept/consent-intercept-config.xml*
Whitelist	Attribute IDs that users should be prompted to consent to	`shibboleth.consent.attribute-release.WhitelistedAttributeIDs`
Blacklist	Attribute IDs that users should **not** be prompted to consent to	`shibboleth.consent.attribute-release.BlacklistedAttributeIDs`
Regex	Attribute IDs matching a regular expression that users should be prompted to consent to	`shibboleth.consent.attribute-release.MatchExpression`

The AttributePredicate determines whether consent should be obtained for an attribute.

Attribute Display Order

Attributes are displayed in the natural order of their IDs. Deployers may wish to customize the order in which attributes are displayed to users, in order to present the most relevant or personal attributes first. The order in which attributes are displayed to users may be customized by providing a list of attribute IDs. Attributes not in the list will still be sorted in their natural order, but subsequent to attributes in the list.

Attribute Display Order List 3.2.0

Attributes are displayed in the order defined by the attribute ID whitelist. If the whitelist is empty or attributes are released which are not present in the whitelist, attributes will be ordered according to the natural order of their IDs, usually alphabetically.

The following example displays the `eduPersonPrincipalName` attribute first and then all other attributes in alphabetic order :

Example custom attribute display order in conf/intercept/consent-intercept-config.xml

```
<util:list id="shibboleth.consent.attribute-release.WhitelistedAttributeIDs">
    <value>eduPersonPrincipalName</value>
</util:list>

<bean id="shibboleth.consent.attribute-release.MatchExpression"
      class="java.util.regex.Pattern" factory-method="compile"
      c:_0="^.*$" />
```

Attribute Display Order List [3.2.1]

To customize the order in which attributes are displayed, define a bean with ID `shibboleth.consent.attribute-release.AttributeDisplayOrder` in *conf/intercept/consent-intercept-config.xml* representing the desired order. The values of the list are attribute IDs.

The following example displays the `mail` attribute first and then all other attributes in alphabetic order :

Example custom attribute display order in conf/intercept/consent-intercept-config.xml

```
<util:list id="shibboleth.consent.attribute-release.AttributeDisplayOrder">
    <value>mail</value>
</util:list>
```

Attribute Display Order Comparator

For advanced customization of the attribute display order, a custom Comparator may be provided. Define a bean with ID `shibboleth.consent.attribute-release.AttributeIDComparator` in *conf/intercept/consent-intercept-config.xml* which implements `Comparator<String>`. For example :

Example custom attribute display comparator in conf/intercept/consent-intercept-config.xml

```
<bean id="shibboleth.consent.attribute-release.CustomAttributeIDComparator"
      class="org.example.CustomAttributeIDComparator" />
```

Attribute Display Name and Values

The names and values of attributes displayed during consent may be customized. By default, the locale-aware attribute display name and display value are displayed.

TODO : Customization

Rejection of Attribute Release Consent

When a user rejects consent to attribute release, an `AttributeReleaseRejected` error page will be displayed. The text presented to the user may be modified via *messages/messages.properties*, see Messages.

Terms Of Use Consent

Consent to terms of use is not enabled by default.

To enable consent to terms of use, you will need to :

1. configure terms of use messages
2. enable the terms of use intercept flow

Configure Terms Of Use Messages

Configure terms of use messages in *messages/messages.properties* (again, see Messages).

The message and title of the terms of use page displayed to users is mapped by the `shibboleth.consent.terms-of-use.Key`, which defaults to the relying party ID. For example :

```
https\://sp.example.org = sp-example-org
sp-example-org.title = Example Terms of Use
sp-example-org.text        = This is an example Terms of Use
```

You may add additional terms of use messages and web page titles specific to relying parties using this mechanism.

Customize Mapping of Relying Party to Terms of Use Message and Title

To customize the mapping of relying party to terms of use message and title, override the `shibboleth.consent.terms-of-use.Key` in *conf/intercept /consent-intercept-config.xml* :

```
<bean id="shibboleth.consent.terms-of-use.Key"
      class="com.google.common.base.Functions" factory-method="compose">
    <constructor-arg name="g">
        <bean class="com.google.common.base.Functions" factory-method="forMap" c:defaultValue="terms-of-use">
            <constructor-arg name="map">
                <map>
                    <entry key="https://sp.example.org/shibboleth" value="example-terms" />
                </map>
            </constructor-arg>
        </bean>
    </constructor-arg>
    <constructor-arg name="f">
        <ref bean="shibboleth.RelyingPartyIdLookup.Simple" />
    </constructor-arg>
</bean>
```

Configure One Terms of Use Message for Every Relying Party

To configure a single terms of use page for every relying party, override shibboleth.consent.terms-of-use.Key in conf/intercept/consent-intercept-config.xml :

```
<bean id="shibboleth.consent.terms-of-use.Key" class="com.google.common.base.Functions" factory-method="
constant">
  <constructor-arg value="my-terms"/>
</bean>
```

and define the terms of use message and title in messages/messages.properties :

```
my-terms = my-tou
my-tou.title = Example Terms of Use
my-tou.text        = This is an example Terms of Use
```

Enabling Terms Of Use Intercept Flow

To enable the terms of use flow, add `terms-of-use` to the post authentication flows in *conf/relying-party.xml.*

For example for use with SAML 2.0 requests, replace :

```
<bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
```

with :

```
<bean parent="SAML2.SSO" p:postAuthenticationFlows="#{ {'terms-of-use', 'attribute-release'} }" />
```

Rejection of Terms of Use Consent

When a user rejects consent to terms of use, a `TermsRejected` error page will be displayed. The text presented to the user may be modified via *messages/messages.properties*, see Messages.

User Interface

The user interface for the attribute release and terms of use consent intercept flows were based on uApprove and are implemented as Velocity templates by *views/intercept/attribute-release.vm* and *views/intercept/terms-of-use.vm*.

Customization

The velocity template can be customized in a similar way to the Login pages.

Localized Messages

Messages displayed to users may be localized in the standard Spring way, for example, by providing *messages/messages_de.properties*.

Revoking Consent

Users may revoke previous consent choices by selecting a checkbox on the login page *views/login.vm*.

The text of the checkbox displayed on the login page is set by the `idp.attribute-release.revoke` property, overridable in *messages/messages.properties*.

Storage

In order to remember users' consent choices and to prompt users to consent to attribute release if attributes change, users' consent choices must be persisted by a storage service. User consent may be stored either client-side (cookies or HTML5 Local Storage [3.2]) or server-side (database). The default is to store consent client-side via cookies.

The storage service used to store consent is configured by the `idp.consent.StorageService` property in *conf/idp.properties*.

See StorageConfiguration.

Limiting the Number of Stored Consent Records

Because cookies provide limited storage capability, the number of stored consent records may be limited. By default, for client-side storage via cookies, the maximum number of stored consent records is 10. Depending on the number of attributes released and whether released attribute values are compared, the default limit of 10 may be increased. At the time of this writing, it is not clear what a reasonable default value should be, but it may be significantly higher. We were conservative with the default.

If server-side storage is used, the number of stored consent records should probably be unlimited, represented by a limit of -1 or 0.

The maximum number of stored consent records is configured via the `idp.consent.maxStoredRecords` property in *conf/idp.properties*.

Limiting Storage Size via Symbolics

Because cookies provide limited storage capability, a map is used to store numbers which refer to attribute IDs in order to reduce the size of consent storage records.

The default mapping of attribute IDs to numbers is defined by the `shibboleth.consent.DefaultAttributeSymbolics` bean in *system/conf/profile-intercept-system.xml*.

Additional mappings of attribute IDs to numbers may be added to the `shibboleth.consent.AttributeSymbolics` bean in *conf/intercept/consent-intercept-config.xml*.

Storage Record Lifetime

The default lifetime for consent storage records is 1 year, and may be configured via the `idp.consent.storageRecordLifetime` in *conf/idp.properties*. When consent storage records expire, they will be deleted by the storage service.

Storage Record Format

The below information is provided for reference but is not a public interface of the system. You may not depend on the format remaining unchanged across even patch releases.

Consent storage records consist of a key and value, similar to other StorageRecords.

The storage key for consent records is a concatenation of the user key and the relying party ID.

The storage value for consent records is the JSON serialized form of a Map of Consent objects.

A per-user index is maintained in order to expire or limit the number of stored records.

```
{
   "jdoe:https://sp1.example.org":{
      "v":"[{\"id\":307},{\"id\":\"mail\"},{\"id\":\"uid\"}]",
      "x":1479847202110
   },
   "jdoe:https://sp2.example.org":{
      "v":"[{\"id\":307},{\"id\":\"mail\"},{\"id\":\"uid\"}]",
      "x":1479847206825
   },
   "jdoe:https://sp3.example.org":{
      "v":"[{\"id\":307},{\"id\":\"mail\"},{\"id\":\"uid\",\"appr\":false}]",
      "x":1479847566214
   },
   "jdoe:_key_idx":{
      "v":"[\"jdoe:https://sp1.example.org\",\"jdoe:https://sp2.example.org\",\"jdoe:https://sp3.example.org\"]"
   }
}
```

Auditing

By default, consent audit logs are written to */logs/idp-consent-audit.log* as defined in *conf/logback.xml*.

The format of consent audit logs are defined by the `shibboleth.consent.attribute-release.AuditFormattingMap` and `shibboleth.consent.terms-of-use.AuditFormattingMap` beans in *conf/intercept/consent-intercept-config.xml*.

IdPAuditFields

ConsentAuditFields

Activation Conditions

Consent flows (like any other profile intercept flow) may be run conditionally based on activation conditions.

Because the attribute release flow has a default activation condition, customizing the activation condition for attribute release will require an AND activation condition to combine the default activation condition with the custom activation condition. By default, the attribute release flow is not activated if both (1) attri butes are not pushed and (2) per attribute consent is enabled. The terms of use flow does not have a default activation condition.

Attribute Checking

Consent flows may be activated based on the presence (or absence) of a particular attribute or value for a user.

The following example activates the attribute release flow if an attribute is present by combining the default activation condition with a custom condition:

Example activation conditon in conf/intercept/profile-intercept.xml

```
<bean id="intercept/attribute-release" parent="shibboleth.consent.AttributeReleaseFlow"
    p:activationCondition-ref="MyCondition" />

<bean id="MyCondition" parent="shibboleth.Conditions.AND">
    <constructor-arg>
        <list>
            <!-- The default condition from system/conf/profile-intercept-system.xml -->
            <bean parent="shibboleth.Conditions.OR">
                <constructor-arg>
                    <bean parent="shibboleth.Conditions.NOT">
                        <constructor-arg value="%{idp.consent.allowPerAttribute:false}" />
                    </bean>
                </constructor-arg>
                <constructor-arg>
                    <bean class="net.shibboleth.idp.saml.profile.config.logic.
IncludeAttributeStatementPredicate" />
                </constructor-arg>
            </bean>
            <!-- A custom condition -->
            <bean class="net.shibboleth.idp.profile.logic.SimpleAttributePredicate" p:useUnfilteredAttributes="
true">
                <property name="attributeValueMap">
                    <map>
                        <entry key="eppn">
                            <list>
                                <value>*</value>
                            </list>
                        </entry>
                    </map>
                </property>
            </bean>
        </list>
    </constructor-arg>
</bean>
```

Reference

Beans

Beans defined in *conf/intercept/consent-intercept-config.xml* :

Bean ID	Type	Function
shibboleth.consent.terms-of-use.Key	Function<ProfileRequestContext,String>	Function which returns the key used to (1) lookup terms of use messages and when concatenated with the user key to (2) lookup storage records, defaults to the relying party ID
shibboleth.consent.attribute-release.WhitelistedAttributeIDs	Set<String>	Whitelist of attribute IDs for which consent should be obtained
shibboleth.consent.attribute-release.BlacklistedAttributeIDs	Set<String>	Blacklist of attribute IDs for which consent should not be obtained
shibboleth.consent.attribute-release.MatchExpression	String	Regular expression matching the attribute IDs for which consent should be obtained
shibboleth.consent.attribute-release.AuditFormattingMap	Map<String,List<String>>	Attribute release audit log format, maps logger name to audit fields
shibboleth.consent.terms-of-use.AuditFormattingMap	Map<String,List<String>>	Terms of use audit log format, maps logger name to audit fields
shibboleth.consent.PreConsentAuditExtractors	Map<String, Function<ProfileRequestContext,Object>>	Audit fields in addition to the default fields populated at the start of the consent flow

shibboleth.consent.ConsentAuditExtractors	Map<String, Function<ProfileRequestContext,Object>>	Audit fields in addition to the default fields used when writing the audit log
shibboleth.consent.AttributeSymbolics	MapFactoryBean	Limits storage record size by mapping attribute IDs to numbers

Properties

Relevant properties defined in *conf/idp.properties* :

Property	Type	Default	Function
idp.consent.StorageService	Bean ID	shibboleth.ClientPersistentStorageService	Name of storage service used to store users' consent choices
idp.consent.userStorageKey	Bean ID	shibboleth.consent.PrincipalConsentStorageKey	**DEPRECATED** Name of function used to return the String storage key representing a user, defaults to the principal name
idp.consent.attribute-release.userStorageKey 3.4	Bean ID	shibboleth.consent.PrincipalConsentStorageKey	Replacement for "idp.consent.userStorageKey" specific to attribute-release flow
idp.consent.terms-of-use.userStorageKey 3.4	Bean ID	shibboleth.consent.PrincipalConsentStorageKey	Replacement for "idp.consent.userStorageKey" specific to terms-of-use flow
idp.consent.userStorageKeyAttribute	String	uid	**DEPRECATED** Attribute whose value is the String storage key representing a user, only used when idp.consent.userStorageKey = shibboleth.consent.AttributeConsentStorageKey
idp.consent.attribute-release.userStorageKeyAttribute 3.4	String	uid	Replacement for "idp.consent.userStorageKeyAttribute" specific to attribute-release flow
idp.consent.terms-of-use.userStorageKeyAttribute 3.4	String	uid	Replacement for "idp.consent.userStorageKeyAttribute" specific to terms-of-use flow
idp.consent.allowDoNotRemember	Boolean	true	Whether not remembering/storing consent is allowed
idp.consent.allowGlobal	Boolean	true	Whether consent to any attribute and to any relying party is allowed
idp.consent.allowPerAttribute	Boolean	false	Whether per-attribute consent is allowed
idp.consent.compareValues	Boolean	false	Whether attribute values and terms of use text are stored and compared for equality
idp.consent.maxStoredRecords	Integer	10	Maximum number of records stored when using space-limited storage (e.g. cookies), 0 = no limit
idp.consent.expandedMaxStoredRecords	Integer	0	Maximum number of records stored when using larger/server-side storage, 0 = no limit
idp.consent.storageRecordLifetime	Duration	P1Y	Time in milliseconds to expire consent storage records

Messages

Relevant messages overridable via *messages/messages.properties* :

Message	Text
no-release.title	Title of error page displayed when attribute release consent is rejected
no-release.message	Text of error page displayed when attribute release consent is rejected
no-terms.title	Title of error page displayed when terms of use consent is rejected
no-terms.message	Text of error page displayed when terms of use consent is rejected

Views

Velocity templates in *views/intercept* :

View	Function
views/intercept/attribute-release.vm	Default attribute release Velocity template

views/intercept/terms-of-use.vm	Default terms of use Velocity template

V2 Compatibility

Although consent in IdPv3 was modeled after the uApprove and uApproveJP plugins to IdPv2, at this time it is not possible to upgrade consent from IdPv2 to IdPv3 because the storage implementations are not compatible.

Notes

TBD