Single Logout in Shibboleth IdP

Important notes on third party cookies

In some browsers, the IFrame-driven front-channel logout doesn't work due to the browser blocking third party cookies.

Every cookie which is sent to a foreign domain via img, iframe, script, etc. tags is considered to be third party, so the session cookie of the SP software in a foreign domain is third party cookie when it is sent in an IFrame. By blocking these cookies, the SP doesn't receive the session cookie and thus it could stop processing the logout request at this point.

Additional links:

- Shibboleth-dev thread on the issue
- How to disable third party cookies in firefox
- Additional explanation in Mozilla Bugzilla
- Same origin policy for cookies
- Further information on third party cookie handling

"Although any third-party cookie restrictions are not a sufficient method to prevent cross-domain user tracking, they prove to be rather efficient in disrupting or impacting the security of some legitimate web site features, most notably certain web gadgets and authentication mechanisms."

Why service providers might need the session cookie

Most of the services do not need the session cookie itself, they only need the NameIdentifier, which is carried by the logout request, so back-channel logout requests are enough for them. But there might be service providers which do not implement back-channel bindings (eg. SimpleSAMLphp), or need front-channel application notification.

Why not fully back-channel?

SAML profiles specification (section 4.4.3.1) clearly states that front-channel should be preferred when sending the logoutrequest to the session authority (IdP). If the user interface is generated by the IdP, it could inform the user about the whole logout process, and each SP response. If the SP would use back-channel logoutrequest, the IdP's response would only contain minimal information (ie. success or failure), and this is not desirable. Also, the IdP would need to execute back-channel requests in parallel and synchronize them with the originating request, so this would make the processing code much more complex.

Technical solution

Our proposal is to prefer back-channel endpoints at the service provider side, unless your application needs to be notified via front-channel. For example,

- if your application behind your SP needs the session cookie with the notification, use only front-channel bindings in the SP metadata,
- otherwise use only back-channel binding in the SP metadata.

By these mutually exclusive endpoint sets, the SP can clearly advise the IdP which binding it should use when contanting this SP. Thus on the IdP side, both implementations need to be available.

Non-technical solution

Another option would be to add a new requirement for your end users. You can claim that banning third-party cookies is unsupported (because it breaks SLO), just like disabling all cookies (which breaks SSO). Convincing your users (and the Shibboleth developers to accept this solution) might be dubious, though.

Features

- Implements SAML2 Single Logout profile
- If initiated by an SP, user must confirm the single logout process: one can choose to logout only from the initiating SP and the IdP.

- Highly customizable front-channel logout interface which leverages javascript and asynchronous operations in order to provide a clean, simple UI.
- UI is usable with javascript disabled.
- Supports localized SP name lookup via Organization elements in SAML metadata .
- Fallback to back-channel logout if front-channel is not supported by the SP.
- Supports Shibboleth SSO sessions (if the SP initiates sessions using Shibboleth1.3 protocol, but supports SAML2 logout).
- Supports full back-channel operation.
- Supports IdP-initiated Single Logout.

UI customization

The UI is located in two JSP files:

- `sloQuestion.jsp` the user chooses whether she wants to logout from all service providers or just from the provider where she came from.
- `sloController.jsp` is the logout UI where every session participant and their corresponding logout status is shown. At the end of the logout process, the user is notified if the single logout was completed.

How it works

SLOServlet

The heart of the logout process is the `SLOServlet`. This servlet is responsible for these actions:

- rendering the logout question and controller page
- initiating front-channel or back-channel logout to one SP (`SLOServlet?action&entityID=...`)
- returning the logout status as a JSON string (`SLOServlet?status`)

With javascript

The controller renders a page where an iframe is placed for every active session participant. This iframe calls the `SLOServlet?action&entityID=...` URL where the logout request is issued for the given session participant. If the request is front-channel, the iframe will make a front-channel SAML message exchange with the peer (using HTTP-Redirect or POST bindings).

The status of the single logout process is queried via asynchronous requests. The status response from `SLOServlet` is a JSON array. This JSON array contains one object with the `entityID` and `logoutStatus` properties for each session participant.

The logout status can be one of the followings:

- `LOGGED_IN`: logout is not initiated for this participant yet.
- `LOGOUT_ATTEMPTED`: logout was initiated.
- `LOGOUT_FAILED`: logout failed.
- `LOGOUT_UNSUPPORTED`: SAML2 logout is not supported by the participant (the metadata does not contain the necessary endpoints).
- `LOGOUT_TIMED_OUT`: timed out waiting for a response.
- `LOGOUT_SUCCEEDED`: logout was successful.

Status queries are issued using exponential backoff timing, until the timeout is reached. Please see the `sloController.jsp` for the exact timing used.

Without javascript

Controller renders an HTML page with `<noscript>` tags. There is one link for each session participant opening up in new window/tab, which can initiate the logout process for that particular SP. Depending on the current logout status, several other controls are enabled on the page:

- `Refresh` button, which will reload the controller HTML with the current status icons to follow the overall logout process.
- `Logout failed` message when logout process was finished, and there was at least one failed session participant.
- `Logout succeeded` message when logout process was finished, and all session participants completed the logout.

IdP-initiated Logout (available since v2.1.3-slo2)

The user can initiate their logout process from the IdP (the URL is `/idp/Logout`). IdP-initiated logout has a clear advantage over SP-initiated logout, because the URL and the UI is fully independent from the current SP software used, thereby providing a unique logout URL for all users of the given IdP.

Non-trivial settings

Security

SAML Single Logout Profile requires the logout requests and responses to be signed or otherwise authenticated. Without this, a user session could be DOS-ed knowing the NameID.

You have two choices

- instruct the SP to sign messages

- configure the IdP not to require authentication of logout messages (and bear with possible DOS-attacks)

Signing messages

Signing can be turned on by setting the **`signing`** property to **`front`** (for front-channel only) or **`true`** in the `ApplicationDefaults` or `ApplicationOverride` element in shibboleth2.xml.

!!! note

```
Signing messages is normally unnecessary for back-channel, as the transport is usually authenticated with the certificates in the metadata. However, for back-channel logout it is the IdP who initiates the HTTP connection to the SP, and it is the **webserver**, who answers the request. Because of the different needs, the webserver almost always uses a different certificate (a server certificate signed by a well-known CA) than the SP (possibly self-signed, client certificate). Therefore the SP must sign back-channel messages as well to authenticate itself to the IdP. Unfortunately, you can only enable signing all (otherwise transport protected) messages, and this may affect performance.
```

Not requiring peer authentication

Message issuer authentication can be turned off by changing the security policy of processing Single Logout messages. You can do this by commenting out the following line from the block **`SAML2SLOSecurityPolicy`** at **`relying-party.xml`**:

```
<security:Rule xsi:type="security:MandatoryMessageAuthentication" />
```

Session lifetime

IdP session lifetime must be longer than any SP session lifetime. Otherwise, if an SP session outlives the IdP session and the user reauthenticates for a new session for other SPs, logout would not terminate session at the first SP.

The IdP can limit the maximum lifetime of the SP session by using the (optional) `SessionNotOnOrAfter` property in the SAML2 authentication statement. SAML1.1 does not have this feature, so **you cannot limit the session lifetime for SPs using Shibboleth SSO protocol.**

> This can be set in the `relying-party.xml` by specifying the number of milliseconds in the **`maximumSPSessionLifetime`** attribute of the **`SAML2SSOProfile`** configuration.

Required changes in the IdP

Name identifier caching in IdP session

In the LogoutRequest the IdP must reference the current user's name identifier. This name identifier is issued as part of the SSO process. In order to efficiently retrieve this information, the IdP should cache the name identifier in the IdP session information object.

Associated ticket: SIDP-336

Session indexing

On receiving a LogoutRequest from a session participant, the IdP must be able to retrieve the IdP session associated with the principal. Session participants use the issued name identifier to identify the principal. The IdP session object can be indexed (and then retrieved of course) by any arbitrary unique key, so we use the name identifier value to index the session.

Associated ticket: SIDP-338

IdP Session invalidation

Currently there is a bug in the IdP implementation which causes the IdP sessions to outlive the session removal.

Associated ticket: SIDP-333

How to use

How to build

- install the maven2 build tool
- source code is available from our git repository
  - you can use the convenient snapshot links below to start playing
  - if you are brave enough, feel free to clone the whole repository and track our development branches (frontchannel-slo for the idp project and slo-configuration

branch for the shibboleth-common project)
- compile the shib-java-common project first with the `mvn -DskipTests install` command (the first build might take quite a long time if you haven't used maven before)
- compile the java-idp project with the same maven command
- install the `java-idp/target/shibboleth-identityprovider-{version}-bin.zip` binary package the same way as you'd install a vanilla Shibboleth IdP bundle

Released versions

- download the latest binary snapshot version from our software distribution site

v2.2.0-slo10

- fix configuration templates
- source code snapshots
  - shibboleth-common-1.2.0-slo2
  - shibboleth-identityprovider-2.2.0-slo10

v2.2.0-slo9

- allow EncryptedID to be used in the initiating request (patch contributed by Michael Simon from Karlsruher Institut für Technologie)
- expose method for programatical back-channel logout
- source code snapshots
  - shibboleth-common-1.2.0-slo2
  - shibboleth-identityprovider-2.2.0-slo9

v2.1.5-slo7

- use AttributeConsumingService/ServiceName to feed the logout interface
- source code snapshots
  - java-shib-common-1.1.4-slo2
  - java-idp-2.1.5-slo7

v2.1.5-slo6

- skip session-indexing under error conditions
- source code snapshots
  - java-shib-common-1.1.4-slo2
  - java-idp-2.1.5-slo6

v2.1.5-slo5

- fixed NullPointerException with non-existent or filtered NameIdentifiers
- fixed a flaw in session-indexing logic, use the whole NameIdentifier as the index, not just the value
- source code snapshots
  - java-shib-common-1.1.4-slo2
  - java-idp-2.1.5-slo5

v2.1.5-slo4

- upstream version bump
  - java-shib-common
  - java-idp

v2.1.4-slo4

- updated Shibboleth-core
- fixed NullPointerException introduced by an erroneous merge in v2.1.4-slo3
  - java-shib-common
  - java-idp

v2.1.3-slo3

- support Terracotta clustering
- source code snapshots
  - java-shib-common
  - java-idp

v2.1.3-slo2

- support IdP initiated logout
- source code snapshots
  - java-shib-common
  - java-idp

v2.1.3-slo1

- support SP initiated front- and back-channel logout

Hints

>

- Don't forget to include Single Logout endpoints in the IdP metadata
- Shibboleth SP prior to 2.1 did not include NameID properly in the LogoutRequest, therefore you cannot initiate SLO with Shibboleth SPs older than 2.1
- Shibboleth SP prior to 2.2.1 answered with Partial logout for back-channel requests due to a bug
- Shibboleth SP (currently released versions) do not distinguish between Success and Partial logout when showing the UI (see this report for details). This is not needed unless you are using back-channel logout.
- If you plan to upgrade a clustered IdP to this version, don't forget to check the new tc-config.xml and rebuild the terracotta boot jar

Missing features

- Administrative logout
- Logout the user in the underlying JAAS provider

Változat #1
document-uploader hozta létre 2025-08-07 12:07:19 CEST
document-uploader frissítette 2025-08-07 12:07:20 CEST