ErrorHandlingConfiguration

Current File(s): *conf/errors.xml, conf/idp.properties, messages/messages.properties, views/error.vm*

Format: Native Spring, Velocity or JSP

Legacy V2 File(s): *error.jsp*

- Overview
- Event Categorization
- Status Mapping
- Servlet Exception Mapping 3.3
- Views and Messages
    - Internationalization
- Reference
    - Beans
    - Properties
- V2 Compatibility
- Notes

Overview

At a high level, there are *events* and *exceptions*.

Most errors that occur in the course of processing requests inside the various web flows are events. Events are a controlled result of processing, while exceptions are more low-level conditions that can't be anticipated (often runtime exceptions that typically result from bugs). The flows defined in the system are designed to deal with events in a predictable way, but exceptions are generally outside the control of the web flow engine and are handled more uniformly as an unrecoverable result.

For back-channel SAML profiles, usually SOAP-based exchanges, events are normally handled by returning SAML error responses or SOAP faults to the requester.

For other profiles, events can be classified as "Local" or not. A Local event interrupts the processing at the IdP, and won't issue a response back to the requesting service (even if this is otherwise possible). When a non-Local event occurs, the IdP will try to return a formal response (e.g., a SAML message) if there's sufficient information available to do so, and if not, it falls back to treating the event as Local.

By default, all events are non-Local unless the configuration specifies they should be Local, which is in keeping with general SAML guidance to try and respond if it's possible to do so.

When issuing a SAML response, the status codes are determined based on the event and the configuration. The status message defaults to very minimal for security reasons, but more detailed messages can be exposed if desired.

The *errors.xml* file controls the mapping of events and exceptions to status codes and to error templates. The default message properties that come with the software provide an optional, but useful, way of indirecting the lookup of detailed error messages through the Spring MessageSource interface, which can help separate messages from the look and feel of errors, and provides some support for internationalization.

In older versions, the default message property files were exposed in multiple files in the **messages** subdirectory and were user-editable. As of V3.3, the default message translations are all system files and a single *messages.properties* file is provided so that specific messages can be overridden, making the addition of messages in newer versions much simpler. Additional translation sets can also be dropped into the **messages** directory if desired.

Event Categorization

Three beans are defined in *errors.xml* to control the classification of events into Local and non-Local, and to control what view templates are used to handle events:

- **shibboleth.DefaultErrorView**
    - Identifies the view name to use by default for Local events. Generally just left alone and set with a property.
- **shibboleth.EventViewMap**
    - A Spring map that links event names to view names. By default, only special local events are overridden, typically events produced by flows that render their output with a view. This illustrates that not all events are "errors".
- **shibboleth.LocalEventMap**
    - A Spring map that enumerates events that should be classified as Local. The map values are flags that determine whether to output audit records when events occur. Usually this is appropriate, but not always.

The default values in these maps are somewhat important for functionality built into the IdP to function as expected, but you are encouraged to add to them as desired.

Status Mapping

The *errors.xml* file also contains Spring map beans that link event names to SAML 1 and SAML 2 Status code lists, as well as a third for SOAP Faults. These maps tell the IdP what status or fault codes to include in its responses based on events that occur. Many are predefined, sufficient for most purposes. You can see the status code lists defined in these mappings in *system/conf/utilities.xml*. In practice, few implementations pay any attention to these finer details. Errors are errors and that's about all that matters.

Servlet Exception Mapping 3.3

While most errors are handled internal to the IdP, if you have external servlets that raise their own exceptions you can map those exceptions back into the error machinery of the IdP by creating mappings in the *web.xml* deployment descriptor.

```
<!-- Send servlet errors through the IdP's MVC error handling. -->
<error-page>
    <exception-type>net.shibboleth.idp.authn.ExternalAuthenticationException</exception-type>
    <location>/profile/MyServlet</location>
</error-page>
```

Views and Messages

Like the rest of the IdP, the user interface for errors is rendered using MVC view templates, which can theoretically use many different kinds of technologies, but by default Velocity and JSP are supported.

As discussed above, it's possible to map events to many different views, and it's actually possible to do the same for exceptions, but this tends to create more work to keep templates consistent in appearance. It's generally easier to use a single template when possible, and the example included with the IdP demonstrates one way of doing this while creating context-sensitive messages. This isn't the only possible way of doing things, but it seems to work well.

The example provided uses Velocity. Using JSP requires renaming or removing the Velocity template from the *views* folder, and restarting the IdP. JSP by design requires that templates appear inside the war file, or unpacked war tree, and should be stored in **edit-webapp/WEB-INF/jsp**.

The *error.vm* example makes use of a Spring feature called a MessageSource, and the files in the **messages** and **system/messages** directories are installed by default as a message source within the IdP. The example does a lookup on an event name or exception class to locate a short key to use, which is then suffixed with ".title" and ".message" to obtain title and message text to insert into the view. This content is output directly with no additional encoding, so you can embed HTML tags in the message data when desired.

If you want to introduce separate view templates for different exceptions, the **idp.errors.exceptionMappings** property can be set to supply a Java Properties bean name that maps exception names to view names.

Internationalization

The Spring message feature used in the example view supports a limited form of internationalization. The browser's Accept-Language header can be used to favor a language-specific message file that will be used in place of the default file. For example, a file named *messages/messages_fr.properties* would be used for a French locale. As far as we know, Spring limits this to a single locale that either has a match or falls back to the default, which does not allow for preference-based selection.

Have a look at the Messages Translation page to find out how to download existing or create new translations of the IdPv3 messages.

Reference

Beans

Bean ID	Type	Default	Function
shibboleth.DefaultErrorView	String	%{idp.errors.defaultView:error}	Identifies the view name to use by default for Local events, generally left alone
shibboleth.EventViewMap	Map<String,String>	See *errors.xml*	Map that links event names to view names, generally used only for specialized flows that use a view to render their final output
shibboleth.LocalEventMap	Map<String, Boolean>	See *errors.xml*	Map that enumerates event names that should be classified as Local
shibboleth.SAML1StatusMappings	Map<String, List<QName>>	See *errors.xml*	Map of non-Local event names to SAML 1 StatusCode values (including substatus values)
shibboleth.SAML2StatusMappings	Map<String, List<String>>	See *errors.xml*	Map of non-Local event names to SAML 2 StatusCode values (including substatus values)
shibboleth.SOAPFaultCodeMappings	Map<String, QName>	See *errors.xml*	Map of non-Local event names to SOAP faultcode

Properties

Properties defined in *idp.properties* directly related to this configuration area follow:

Property	Type	Default	Function
idp.errors.detailed	Boolean	false	Whether to expose detailed error causes in status information provided to outside parties

idp.errors.signed	Boolean	true	Whether to digitally sign error responses in SAML or similar protocols, if signing is otherwise warranted (this can prevent a simple denial of service vector, since errors are simple to trigger)
idp.errors. defaultView	String	error	The default view name to render for exceptions and events
idp.errors. exceptionMappings 3.2	Properties		A property set mapping exception class names to error views. The matching by class name does not support wildcards, but does do substring matches (so it's not necessary to fully qualify the class).
idp.errors. excludedExceptions 3.2	Collection <Class>		Exception classes to ignore (causing them to bubble outward, so use with caution)

V2 Compatibility

There is no direct compatibility with V2 error handling and the older interfaces to error information are not compatible.

Notes

TBD