Fediz Extensions

Fediz Extensions

This page describes the extension points in Fediz to enrich its functionality further.

Callback Handler

The Sign-In request to the IDP contains several parameters to customize the sign in process. Some parameters are configured statically in the Fediz configuration file, some others can be resolved at runtime when the initial request is received by the Fediz plugin.

Configuration values common to both WS-Federation and SAML SSO

The following table gives an overview of the parameters which can be resolved at runtime for either the WS-Federation or SAML SSO protocols.

XML element	Callback class
issuer	IDPCallback
logoutRedirectToConstraint	ReplyConstraintCallback

WS-Federation

The following table gives an overview of the parameters which can be resolved at runtime for the WS-Federation protocol. It contains the XML element name of the Fediz configuration file, the query parameter name of the sign-in request to the IDP as well as the Callback class.

XML element	Query parameter	Callback class
authenticationType	wauth	WAuthCallback
homeRealm	whr	HomeRealmCallback

freshness	wfresh	FreshnessCallback
realm	wtrealm	RealmCallback
signInQuery	any	SignInQueryCallback
signOutQuery	any	SignOutQueryCallback
request	wreq	WReqCallback
reply	wreply	ReplyCallback

If you configure a class which implements the interface `javax.security.auth.callback.CallbackHandler` you get the corresponding Callback object where you must set the value which is then added to the query parameter. The Callback object provides the `HttpServletRequest` object which might give you the required information to resolve the value.

Here is a snippet of the configuration to configure a CallbackHandler:

```
...
        <protocol xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
type="federationProtocolType" version="1.2">
            ...
            <homeRealm type="Class" value="MyCallbackHandler " />
            ...
        </protocol>
...
```

And a sample implementation of the CallbackHandler:

```
public class MyCallbackHandler implements CallbackHandler {

    public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof HomeRealmCallback) {
                HomeRealmCallback callback = (HomeRealmCallback) callbacks[i];
                HttpServletRequest request = callback.getRequest();
                String homeRealm = ...
                callback.setHomeRealm(homeRealm);
            } else {
                throw new UnsupportedCallbackException(callbacks[i],
"Unrecognized Callback");
            }
        }
    }
}
```

Custom Token Validator

It is possible to plug in a custom Token Validator for either protocol as well using the "tokenValidators" configuration parameter. This takes a list of Strings, each of which correspond to the class name of a TokenValidator instance. For example:

```
...
        <protocol xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
type="federationProtocolType" version="1.2">
            ...
            <tokenValidators>
                <validator>org.apache.cxf.fediz.core.federation.
CustomValidator</validator>
            </tokenValidators>
            ...
        </protocol>
...
```