HP OpenView Select Federation

For the HP-UX, Linux, Solaris and Windows® Operating Systems

Software Version: 6.60

Web Application Developer’s Guide

Document Release Date: November 2006
Software Release Date: November 2006

Documentation Updates

This manual’s title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP OpenView Support web site at:

www.hp.com/managementsoftware/support

HP OpenView online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1 Introducing the HP OpenView Select Federation Web Application Developer's Guide

This *HP OpenView Select Federation Web Application Developer's Guide* describes how to use the Select Federation Software Developer's Kit (SDK) to add federation functionality to web applications. This document also describes how to use the SDK to integrate an Authority (IDP) installation of Select Federation with back-end data sources and various authentication systems. Also described in this document is how the SDK sample code is laid out on the CD, how to build the samples and what the samples demonstrate.

This guide is intended for developers wanting to integrate Select Federation into applications, into an existing identity management deployment, or with an identity management product.

The Select Federation SDK consists of the following:

- This *OpenView Select Federation Web Application Developer's Guide*.
- *HP OpenView Select Federation Web Services Developer's Guide*, which describes how to develop and deploy new web services, as well as provides samples that you can build and use as a basis for your own development efforts.
- API documentation in the `<`*`cd-base-directory`*`>/docs/api/index.html` file.
- Web Application API samples in the `<`*`cd-base-directory`*`>/api/samples/` directory.
- Web Services samples in the `<`*`cd-base-directory`*`>/web-services/filters/samples/` and `<`*`cd-base-directory`*`>/web-services/api/samples/` directories
- Filter samples in the `<`*`cd-base-directory`*`>/filters/samples/` directory.

SDK Requirements

Prerequisites

This guide assumes a working knowledge of the following:

- Identity Management
- Federated Identity
- Select Federation Architecture (see the "Select Federation Architecture" chapter in the *HP OpenView Select Federation Configuration and Administration Guide*.)
- The basics of popular web application development technologies such as HTTP, HTML, Java Server Pages and CGI scripts

System Requirements

Select Federation is designed to work with a number of hardware and operating systems configurations. The flexibility inherent in Select Federation extends to the third-party applications that it supports, namely the application servers, database servers, and LDAP servers. See "System Requirements" in the *HP OpenView Select Federation Configuration and Administration Guide* for the specific hardware and software system requirements.

Overview

The following sections provide an overview of the functionality that can or should be added to enable applications for a federation. These sections also introduce the various ways to create this functionality offered by Select Federation and its SDK.

- Federation Functionality
- Enabling Applications
- Enabling Authorities

Federation Functionality

In its essence a *federation* is established between partners, more than between accounts. The partners agree that one (the Authority role) will issue *assertions* about users to the other partner (the Application role). A single installation can act in both roles at various times, towards various partners and/or for different users. However it is not very common for a single installation to play both an Application and an Authority role.

A particular user is federated with the Application when the Authority has established an identifier for that user to the Application, such as when the Authority and Application share an identifier for the user. When this identifier is *persistent* the Authority will provide the Application with the same identifier each time the user visits the Application (and authenticates at the Authority). The popular federation protocol specifications strongly recommend that Authorities provide distinct Applications (actually distinct partners) with different identifiers for the same user, so called *pseudonyms* for user privacy reasons. It is also possible that the identifier for the same user to the same Application is different for each visit — the identifier is a *one-time pseudonym*.

Whereas almost all Authorities have a need to manage federations, typically only some Applications have this need. The best way to enable an Application depends to some degree on its need to manage federations. There are a few core aspects to managing a federation; here only brief introductions are provided, for details it is recommended to read the various SAML and Liberty Alliance ID-FF specifications and guideline documents.

It is important to realize that it is always the Authority that is responsible for the user federation. The Authority creates and issues the identifier. An Application asks for a user identifier, and may indicate to the IDP whether a federation exists. The IDP can then create a federation or not.

An Application that has obtained a federated identifier for a user from an Authority can then ask that Authority to perform certain operations, such as the following:

- Log the user out from all Applications that the user is logged into. This is also known as *global logout* or *Single LogOut* (SLO).

- Terminate the federation, which means remove the persistent pseudonym from its record for the user.

Likewise the Authority can ask the Application to logout the user and can inform the Application that it terminated the federation.

Account Linking

When an Application already has local user accounts it is common practice to link accounts with the Authority. This means that the Application links its local user identifier to the federated identifier provided by the Authority. Select Federation is designed to take care of this mapping between the federated identifier and the local identifier. If needed, all the Application needs to do is to provide Select Federation with the local user identifier for the authenticated user. If the Application does not use local user accounts at all (which is possible in a federated deployment), or does not have local user identifiers that were established before a federation was taken into use, it may leave it up to Select Federation to auto-generate local user identifiers.

Some Common Scenarios

The principles and concepts in federation can be illustrated with a few common scenarios. It is possible to play these scenarios with the `sf-demo`, which ships with Select Federation and is installed by default. Playing these scenarios requires two different installations of Select Federation where one acts as an Authority and the other as an Application. For the best experience it is important that the Application have a local directory with local user accounts. See Chapter 2, Creating a Test Setup for more information.

The following sections describe some common scenarios.

User Visits Application Directly

The user browses directly to the Application. The Application needs to authenticate the user but as the user is not yet authenticated the Application cannot know if the user should authenticate with a local account or through a partner IDP. Therefore, the Application provides links for both possibilities.

In the `sf-demo` the two possibilities for authenticating a user are shown as follows:

- With a local account: `Login locally to demo SP application`
- Through a partner IDP: `Login via IDP` (where `IDP` is replaced by the configured friendly name of the Authority partner or partners).

Locally Login to the Application

The Application logs the user in using whatever method it prefers. The `sf-demo` application actually uses either Select Access (if that is in use) or the local Select Federation installation's IDP functionality to do this. It is now possible for the Application to offer the user the possibility to "link account with a partner" or "initiate federation", but these options are somewhat artificial as they do not serve a clear purpose for either the user or the Application at this point.

Login to the Application Through the Authority

The Application asks the authority to authenticate the user. This typically happens through HTTP redirects. If the user was already authenticated to the Authority, the Authority may be able to send the user back to the Application with the necessary assertions conveyed along. If the user was not yet authenticated, the Authority may prompt the user for credentials.

It is also possible that the Authority prompts the user for consent to establish the federation or to release personal information. In the end the user is directed back to the Application which now has received a federated name identifier. If it has not yet mapped the federated identifier with the user, that is, not yet linked accounts, it can choose to do so now, a process that is called *activation* (see Chapter 4, Activation and Auto Enrollment for details). If the local user name is available, the Application can establish the mapping silently. Otherwise the Application will need to ask the user for the user's local user ID (and probably for the local password).

Login to the Authority First

It is also quite common for users to login to authorities before visiting any federated Application. A typical example is an employer portal. When the user visits the portal, the user is authenticated locally to that portal. In the `sf-demo` this is represented by "Login locally to demo IDP portal". The user is then offered links to the various partner Applications. When the user clicks such a link the IDP will construct an assertion and then direct (or redirect) the user to the Application. In the process the IDP may ask the user for consent to establish the federation or to release personal information. The Application will now receive the federated identifier, but if available. the Select Federation installation at the Application site will provided the mapped local identifier.

Single Sign-On

Single-Sign-On (SSO) simply occurs when a user visits a second Application that in turn request authentication from the same Authority where the user was recently authenticated. The Authority then may have no reason to prompt the user again for the user's credentials.

Application-Initiated Single LogOut

Application-initiated single LogOut occurs when the Application offers the user an option that causes the Application to request the Authority to initiate a global logout process. The authority will notify all Applications to which the user was logged in that the user wishes to log out. Note that this happens on the basis of the authenticated session between the user and the Authority that was used to assert the user to the initiating Application. It is possible that the user has multiple sessions with the Authority, in which case only the Applications that belong to the session used for the initiating Application will be notified.

Authority-Initiated Single LogOut

The Authority (portal) offers the user an option that causes it to notify all Applications (the ones the user had logged into through this Authority) to log the user out.

Terminate Federation

Both an Application and an Authority can ask the other party to terminate a federation. This is rarely available to end users who do not need to terminate a federation. The `sf-demo` has links for this to enable testers to remove and re-establish federations.

Now that the federation concepts are somewhat familiar, the following sections describe what it means to enable Applications and Authorities.

Enabling Applications

An Application may need any combination of the following aspects of user information:

- user ID
- user is authenticated
- one or more user attributes, such as name, email address, age, and so on
- partners that *could* authenticate the user
- information on how and when the user was authenticated
- information on how to invoke a service for that user, such as the possibility to write to the user's calendar service

The first three items are common for all applications whether or not they are deployed in a federated environment, whereas the last three items are more typical for applications that are deployed in a federated environment.

For many applications access control is important and then it may be useful to think of *protecting* an application. Protection often means obtaining a certain combination of the above snippets of user information and then enforcing a decision. For example an application may require that users are known (have a user id that is recognized) and that certain user attributes meet certain criteria. It is quite common for applications to use user attributes as inputs for application specific policies.

As an example one can think of an application that requires a user to authenticate with an X509 (client) certificate. The certificate will state the user ID and the party that issued the certification (authenticated) and may contain some attributes about the user. Application servers can make the information that is in the certificate available to applications using APIs that are specific to the application server and the programming language used to develop the application.

Select Federation and the Select Federation SDK offer various ways to protect applications as well as provide user information to applications. The best way to enable a specific application depends on the needs of the application as well the deployment environment and application server.

Enabling Applications Through Integration with HP OpenView Select Access

Select Federation can be integrated with Select Access and then Select Access can be used to protect applications. Select Access can provide the user ID and user attributes to applications and can be configured to enforce authentication and attribute criteria. In general, Select Access can be used to enforce fine-grained access control to particular parts of an application. Using Select Access requires little or no changes to existing applications.

Select Access has what are called **Enforcers** for a large variety of web and application servers. An Enforcer ensures that a required policy is enforced. In practice, the policy is that only authenticated users can access certain resources. See the *HP OpenView Select Access* documentation for details.

When Select Federation is integrated with Select Access, users that are visiting from another domain, such as those users that are authenticated by a partner Authority, are essentially copied into the directory that is used by Select Access, including any user attributes that were provided by, or obtained from, the Authority. Select Access then treats those users as any other user in its directory.

Integration with Select Access is an effective approach when one or more of the following criteria are met:

- Select Access is already in use.
- The application needs fine-grained, centrally managed access control.
- The application serves a high number of internal users and only few visiting users.

Enabling Applications with Select Federation Filters

The Select Federation SDK offers *filters* that can be used to protect applications effectively without requiring changes to application code. In this model a filter is installed at the application server. Applications that only need to be protected while being accessible to federated users will not need any changes.

The filter communicates with the local Select Federation installation (with the "Application" role). Select Federation currently has filters for IIS and Apache web servers, but it is possible to build filters for other environments. The filter is configured to take certain actions when requested URLs match a configured pattern. Typically, the configuration is set up to request authentication from an Authority for certain URLs (this protects the application) and to provide certain user attributes to the application. In addition filters can provide the application with information about the authentication event. Filters provide user information by adding it to the (http) request in a manner that is consistent with the application development model used by the application server.

The use of Select Federation filters is especially effective when one or more of the following criteria are met:

- Select Access is not used in the deployment environment.
- The application is deployed on IIS or Apache and is not J2EE based.
- The application does not have a need for centrally-managed fine grained access control.
- The application primarily serves visiting users and not local users.

See the *HP OpenView Select Federation Configuration and Administration Guide* for detailed information on filters.

Enabling Applications with the Select Federation SDK APIs

The Select Federation SDK offers a Java Interface that can be implemented by Java applications or Java libraries (APIs) that can be called by Java applications. The Event Plugin Interface enables applications to listen and react to federation events (see SP Event Plugin Interface on page 20 for more information). The APIs, however, require the application to be more proactive in calling Select Federation. The use of APIs requires significant changes to applications. However, it is typical for applications to limit the of use of the Select Federation APIs to pages for identity management, such as pages for login, logout, user registration, and so on. The APIs are used to update the application-specific representation of the user and the core of the application operates with that user representation (as before).

Select Federation APIs are effective when the following criteria are met:

- The application is written with Java-based technologies such as servlets or Java Script Pages.
- The application needs a high level of control over federated users.

See Chapter 3, Select Federation API Overview for API overview information. For complete details on the Select Federation SPAPIs and IDPAPIs, see the `<cd-base-directory>/docs/api/index.html` file on the Select Federation SDK CD.

Enabling Authorities

An Authority or Identity Provider, must authenticate users and provide information about those users. Select Federation can be set up to work with various user directories and comes with a minimal page to collect credentials from end users. As for Applications there are a few possibilities for building Authority deployments.

However, unlike applications, an authority is more closely integrated with the Select Federation installation. Most of the tasks of an Authority are taken care of by Select Federation and completion of an Authority deployment may involve very little development of actual applications (web pages). Often it is sufficient to ensure proper branding of end user facing pages and construct a good looking login page. See the "Branding the End User Pages" section in the *HP OpenView Select Federation Configuration and Administration Guide* for more information.

Authority Relies on Select Access

This option is supported by Select Federation and does not require the Select Federation SDK. It is also the preferred solution for Authorities that use non-Java web servers to present end user pages. In this case, Select Access authenticates the users.

If the Authority offers pages to end users, that is, acts as a kind of portal, then the Select Federation Application Helper applications can be used to construct single sign-on (SSO) URLs to the Application partners. See the "Using the Application Helper" section in the *HP OpenView Select Federation Configuration and Administration Guide*.

The Authority uses the built-in support for LDAP directories to obtain attributes about users.

Authority Uses SDK API to Authenticate Users and Portal Pages

An Authority that does not rely on Select Access for authentication needs to provide the means for end users to authenticate, which often means showing a login page. The login page collects credentials and Select Federation verifies these credentials against the configured directory. See the description of the `web/login.jsp` file in IDP-Sample on page 28.

Alternatively, the Authority can make use of a custom-made *Authentication plugin* (see IDP Authentication Plugin Interface on page 20 for more information). Such a plugin can be written to support virtually any method for user authentication. Not only can such a plugin verify credentials against any back-end system, it can also interact with the user as required. For a detailed description of the `IDP-SampleAuthnPlugin`, see the `<cd-base-directory>/docs/api/index.html` file on the Select Federation CD.

The Select Federation SDK contains an IDPAPI that can be used to obtain URLs for single sign-on to Partner Applications, for Single LogOut, and for federation termination. See IDP-Sample on page 28 for a description of using the IDPAPI. For an overview of the IDPAPI, see IDP API on page 20.

The Select Federation SDK also enables development of a *Directory plugin*, which can be used to obtain user attributes from sources other than an LDAP directory or relational database. See IDP-AuthDir on page 32 for a description of using the `IDPAuthnPlugin_Dir` plugin. For an overview of the IDPAPI, see IDP API on page 20.

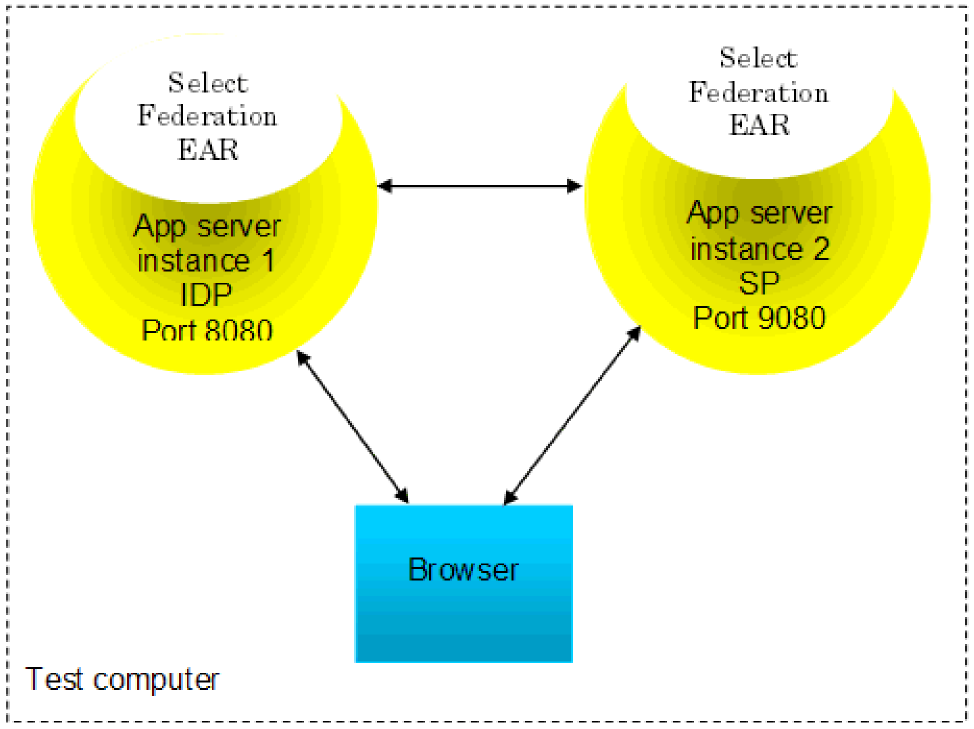
Other user-related services can be exposed as standards-based identity web services. Examples of such services include: geolocation, calendar, and so on. See the *HP OpenView Select Federation Web Services Developer's Guide* for more information.

2 Creating a Test Setup

As a developer, you may want to setup a test environment, where you run both the Identity Provider (IDP or Authority) site as well as the Service Provider (SP or Application) site. In such a case, you need to set up two instances of your application server. If you would like to run both instances on the same computer, make sure the application servers are configured to run on different ports.

The following figure shows a typical test layout:

Figure 1 Typical Test Layout


Select Federation EAR
App server instance 1 IDP Port 8080
Select Federation EAR
App server instance 2 SP Port 9080
Browser
Test computer

Internet Explorer has a bug which confuses redirects between two web servers running on the same computer. If your development setup is on the same computer, please make sure you address the Identity Provider (IDP) with a different host name than the Service Provider (SP). One easy way to do this is to refer to the IDP by IP address and to the SP by the DNS name of the computer. If your browser is also on the same computer, you can refer to the IDP as `localhost` and the SP as `127.0.0.1`. **Be sure to use this terminology when you configure the two Select Federation instances**.

Follow the steps in Chapter 4, “Setting Up Partnerships” and Chapter 5, “Managing Partners” of the *HP OpenView Select Federation Configuration and Administration Guide* to set up a federation between a Select Federation instance and another site. In the case of a test setup, these steps should be followed for both the federation instances that you are setting up. The IDP and SP instances created in this test setup will be useful for deploying and testing the samples.

Several samples as well as filters require attributes of authenticated users to be available at the SP application. For a detailed description of Attribute policy configuration, see the “Configuring the Attribute Policy” section in the *HP OpenView Select Federation Configuration and Administration Guide*. Be sure to follow this configuration so that the required attributes are available to the SP.

3 Select Federation API Overview

This chapter describes how to access the Select Federation API documentation and provides a brief overview of the Select Federation APIs. For descriptions of the API samples that are on the Select Federation CD, see Chapter 5, API Samples.

API Documentation

The Select Federation SDK CD contains detailed API documentation of the APIs. The documentation is available at:

```
<cd-base-directory>/docs/api/index.html
```

Each sample on the Select Federation SDK CD, located in the `<cd-base-directory>/api/samples/` directory, demonstrates the use of one aspect of the Select Federation APIs. See Chapter 5, API Samples for descriptions of each sample on the CD.

Select Federation Main APIs

Select Federation includes the following three main APIs that can be used by developers:

- Plugin APIs
- IDP API
- SP API

The following sections describe each API.

Plugin APIs

The Plugin APIs are for tighter integration with existing infrastructure components such as Identity Management Systems or applications. Following are the available plugin APIs, which are described in the respective sections:

- IDP Authentication Plugin Interface
- IDP Directory Plugin Interface
- SP Event Plugin Interface

See API Documentation on page 19 for information on accessing the API documentation and the Chapter 5, API Samples for descriptions of the samples on the Select Federation CD.

IDP Authentication Plugin Interface

The IDP Authentication Plugin interface enables a customer to provide a tightly-integrated authentication capability that does not require redirects to the IDPAPI. This makes it easy for Select Federation to integrate with off-the-shelf Identity Management Systems.

IDP Directory Plugin Interface

The IDP Directory Plugin Interface enables Select Federation acting as an Authority or IDP to obtain profile attributes for a user. Select Federation supplies an LDAP and a file-based implementation of this Interface, but customers can write their own (for example, to integrate with an RDBMS).

SP Event Plugin Interface

The SP Event Plugin Interface enables customers to "listen" to federation events at a Service Provider (SP) installation. When Select Federation acts as an SP and an event plugin is configured, Select Federation calls the login method of the configured event plugin when it receives an IDP authenticated user. The login method is called after all configured attributes about the user have been received from the IDP.

Similarly, when a user does a global logout, and Select Federation acts as an SP, it receives the global logout event (either on the front-channel through a browser redirect / GET, or through the backchannel) and calls the logout method of the configured event plugin.

In addition, when a user terminates a federation at the SP, the deactivate method is called. The plugin implementer can use the SPAPI to get more information about the current user session as demonstrated in sp-SampleEventPlugin on page 38.

IDP API

The IDP API is a simple way for integrating Select Federation into an existing deployment of an Identity Management System.

See API Documentation on page 19 for information on accessing the API documentation and the Chapter 5, API Samples for descriptions of the samples on the Select Federation CD.

SP API

The SP API is a simple way in which an SP side Application can integrate with Select Federation. An Application may use the Select Federation provided filter, and in addition to that use the SPAPI to gain more information about the user.

J2EE Access Filter

Select Federation SPAPI includes a versatile Access Filter that can allow J2EE applications to integrate with itself without modifying any code. The Access Filter is a J2EE Servlet filter and needs to be included in the application's deployment descriptor, the `web.xml` file.

The access filter uses the Select Federation SPAPI to obtain user identity information and to request authentication for unauthenticated users. This section describes how to use and configure the Select Federation J2EE Access Filter.

How to Install the J2EE Access Filter

The Access Filter is a servlet filter and is included in the deployment descriptor of the customer's J2EE web application that is typically comprised of JSPs and servlets. You may enable the servlet filter for any or all URLs represented by the application, by following the configuration steps listed in the next section. No installation is required.

How to Configure the J2EE Access Filter

To configure the J2EE Access Filter, open the deployment descriptor file, `web.xml`, of your deployed application and perform the following steps:

1 Specify the location of the Access Filter:

```
<filter-class>com.trustgenix.tfsSPAPI.AccessFilter</filter-class>
```

2 Specify which URLs you want to protect within the URL space of your application.

   For example, you may want to protect a user's personalized page that has information that only the user should see.

3 Specify any of the following configuration parameters that are required by the specific deployed application:

Parameter Name	**Default Values**	**Description**
`excludePrefixes`	None	URL prefixes that are excluded from authentication.
`protectedPrefixes`	/*	URLs that require the user to be authenticated.
`passivePrefixes`	None	URLs for which passive authentication of the user is attempted.
`loginURL`	None	If present, the Access Filter redirects unauthenticated users attempting to access protected URLs to this URL
`activateURL`	None	If present, users who are visiting the Select Federation site for the first time from an IDP are redirected to this URL
`ssoIDP`	None	The IDP to be used for authenticating unauthenticated users. If this variable is defined, the user is always navigated to the specified IDP. If this variable is set to the value `default`, the default IDP configured with Select Federation is used.

Parameter Name	Default Values	Description
`ssoFederate`	True	If this flag is set, the SSO request to the IDP requests the IDP to create a new federation if one does not exist already. This applies only when the naming policy on the IDP is either `pseudonym` or `anonymous`.
`ssoAnonymous`	False	If this flag is set, the SSO request is for anonymous authentication. If the IDP permits anonymous authentication, it will send a new federated identifier each time.
`minAuthnContextRef`	None	If this is set, the SSO request contains a specific URI for an *Authentication Context*, such as password, PKI, or Mobile Contract. If this is not set, the IDP uses its default Authentication Context, which is sufficient in most cases.

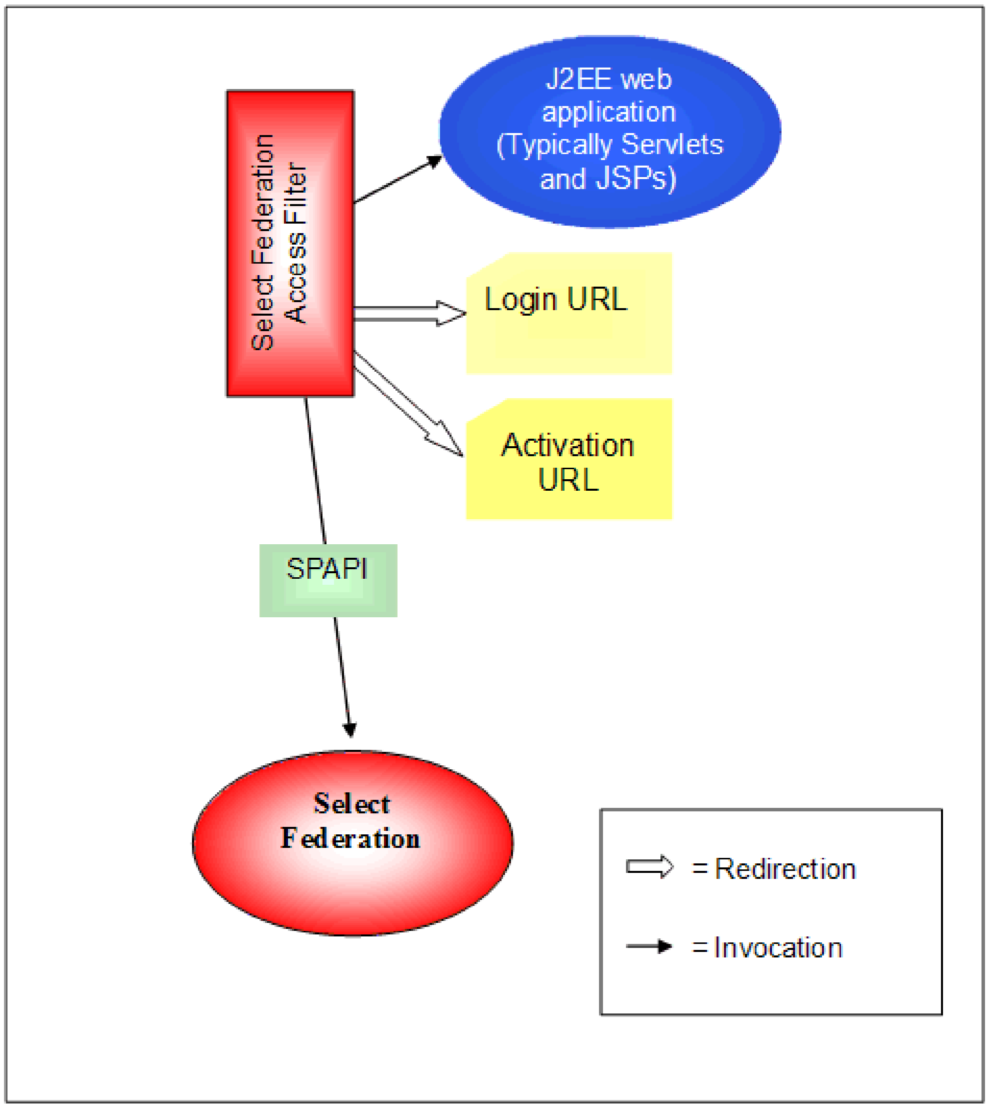
➤ The Access filter maintains a session attribute called `spUser` in the J2EE session of the user. If this attribute is not set, the user is not expected to be logged in.

How to Use the J2EE Access Filter

The following code snippet and figure illustrate a sample of using the J2EE access filter. Note that the J2EE `<filter-mapping>` directive enables the filter for the application's protected URL space:

```
<filter>
   <filter-name>AccessFilter</filter-name>
   <filter-class>com.trustgenix.tfsSPAPI.AccessFilter</filter-class>
   <init-param>
      <param-name>loginURL</param-name>
      <param-value>/protected/login.jsp</param-value>
   </init-param>
   <init-param>
      <param-name>activateURL</param-name>
      <param-value>/protected/activate.jsp</param-value>
   </init-param>
   …
</filter>
<filter-mapping>
    <filter-name>AccessFilter</filter-name>
    <url-pattern>/protected/*</url-pattern>
</filter-mapping>
```

Figure 2 Sample Usage of the J2EE Access Filter


Select Federation Access Filter
J2EE web application (Typically Servlets and JSPs)
Login URL
Activation URL
SPAPI
Select Federation
= Redirection
= Invocation

See API Documentation on page 19 for information on accessing the API documentation and the Chapter 5, API Samples for descriptions of the samples on the Select Federation CD.

4 Activation and Auto Enrollment

Select Federation provides a unique feature beyond simple federated single sign-on (SSO) called New User Activation or simply Activation. The basic idea is to bridge the gap which occurs when a user at the Identity Provider or SAML producer visits a Service Provider or SAML Consumer site for the first time, but that site does not recognize the user since the user is visiting for the first time.

Select Federation creates a new federated account for the user which does not have a local user ID associated with it. Select Federation then fetches the user's activation attributes from the SAML Producer or IDP using either the Liberty Personal or Employee Profile Services or using the SAML Attribute Authority Service.

Select Federation also has an optimal mode where these attributes are automatically pushed to the SP or SAML Consumer with the authentication assertion. The usage of these attributes depends upon how your application integrates with Select Federation.

Using the J2EE Access Filter

If you are using the Access Filter provided as a part of Select Federation, it navigates the user to a special Activation URL, which is specified by the application to the filter as an initialization parameter. The attributes retrieved as a part of the activation process are available to the application through the Select Federation federation session `TFSSessionInfo` object, which is stored in the servlet request object. You can use the following code to retrieve this object:

```
TFSSessionInfo tfsSessionInfol =
(TFSessionInfo) request.getAttribute ("tfsSessionInfo");
```

The application is expected to use the `getProfile()` method of the `TFSSessionInfo` object to obtain an object of type Map which contains the various activation attributes as defined in the Select Federation `tfsconfig.properties` file.

Using the Select Federation SPAPI

If you are not using the Access Filter and using the SPAPI directly, when you call the method `LocateTFSSession` in the Select Federation SPAPI, this method throws an Activation Exception if the user is a new user who does not have an associated local user ID. The `TFSSessionInfo` object is a public member of this exception class. As in the Access Filter case, the application is expected to obtain the map of fetched activation attribute name-value pairs by calling the `getProfile()` method.

Setting the Local User ID

Once you have decided that the user needs to be stored locally, you may call the method `SPAPI:updateLocalUserId()` to set the user's local ID. Once this ID is set, the activation exception will not be thrown for this user. It is not necessary to set this local user ID, but it helps to not get the Activation Exception each time the user returns to your site.

Configuring the Activation Attributes

See Chapter 10, "Configuring Attributes" in the *HP OpenView Select Federation Configuration and Administration Guide*.

5 API Samples

This chapter provides descriptions of the web application samples provided on the Select Federation CD, to help you understand the capabilities of Select Federation and to easily mimic the configurations for your own federations.

Samples list

The web application API samples on the Select Federation SDK CD are in the `<cd-base-directory>/api/samples/` directory.

The following samples are included on the CD:

- IDP-Sample: Demonstrates the use of IDPAPI.
- SP-Sample: Demonstrates the use of the SPAPI.
- IDP-SampleDirPlugin: Demonstrates how a Directory plugin can be written. The sample provides the sources for equivalent functionality as the `DirPlugin` file, which ships as a part of Select Federation.
- IDP-SampleAuthnPlugin: Demonstrates the use of the AuthnPlugin interface in the IDPPlugin API.
- IDP-AuthDir: Demonstrates how to use `IDPAuthnPlugin_Dir`, an Authentication plugin shipped with Select Federation.
- SP-Activation: Demonstrates the use of the built in Access Filter and the use of the SPAPI.
- IDP-Portal: Demonstrates IDP initiated single sign-on.
- IDP-Intro: Demonstrates how to find out the IDP that a user belongs to.
- SP-Intro: Demonstrates the use of Liberty `ID-FF` introduction cookies.
- IDP-Proxy: Demonstrates the use of the proxy function for authentication at an SO using the "home IDP" of a user that is not trusted by the SP.
- sp-SampleEventPlugin: Demonstrates the use of `sp-EventPlugin` to allow applications to be either called or redirected to when a user logs in, logs out or the user's federation is terminated.

➤ The web services API samples are located in the `<cd-base-directory>/web-services/filters/samples/` and `<cd-base-directory>/web-services/api/samples/` directories. For more information about the web services APIs and these samples, see the *HP OpenView Select Federation Web Services Developer's Guide.*

Building the Samples

You can build the samples by copying them from the Select Federation CD to a location on your hard disk.

Required Software

- **Ant**: You need the `Ant` tool from the Apache Jakarta project. Ant version 1.5 is desirable, though earlier versions may also work.
- **JDK**: You will need JDK version 1.4.2 or later.
- **J2EE Servlet Engine**: Since all the samples involve servlets or JSPs. You need a J2EE servlet engine (such as. Tomcat, IBM WebSphere, BEA WebLogic, and so on). The sample applications can reside on the same server as Select Federation.

Build Process

To build a sample, change your current working directory to the top-level directory of the sample, for example.:

```
cd samples/idp-authnplugin
```

At the top-level directory enter the following command:

```
ant clean package
```

The output of the compilation command will be in the directory named `dist`.

Samples Description

➤ Any code using `authnplugin`, `dirplugin` or `eventplugin` APIs must be run on the same J2EE server as Select Federation.

IDP-Sample

This sample is a basic application that demonstrates the use of the IDPAPI in a simple Application scenario. It uses the IDPAPI to register a locally logged in user.

Sources

- **`web/login.jsp`**: This is the login page for users being redirected from the IDP. Set the URL to this page as the `LoginURL` property in the `tfsconfig.properties` file. This logs in the user and registers the user ID locally.
- **`web/index.jsp`**: This is a page that the user will see if the user goes directly to the `idp-sample` (without going through an SP). This page uses the IDPAPI to show federation termination URLS with all SPs the user is federated with.

- **web/federate.jsp**: This page is used if you set its URL to the `consentURL` property in `tfsconfig.properties`. Setting the `consentURL` property means that the user is asked whether to federate the user's account with a particular SP when establishing the federation.

Running the IDP-Sample Sample

To run the `idp-sample`, perform the following steps:

1 Configure the `loginURL` and optionally the `consentURL` properties in the `tfsconfig.properties`.

  These properties must be set to the URLs corresponding to `login.jsp` and `federate.jsp` of the sample. If you do not want to ask the user's permission to federate, you should omit adding the consentURL property to the `tfsconfig.properties` file.

  ➤ You need to restart Select Federation after you have edited the `tfsconfig.properties` file.

2 Deploy the `idp-sample.war` to the J2EE application server.

3 Go to a properly configured SP and attempt a federation.

  You are redirected to the IDP's login page (`login.jsp`).

4 Enter any username and password combination to login in at the IDP.

  If you set the consentURL, the consent page (`federate.jsp`) opens. If not, you are redirected back to the SP with a successful federation.

  You may navigate to the `index.jsp` of the sample at any time to see the logged in user name, other active SP logins and the ability to terminate federation and logout globally.

SP-Sample

This sample is a basic application that demonstrates the use of the SPAPI in a simple Application scenario, without using the Access Filter. It does single sign-on, federation, federation termination, global logout and activation. It uses the SPAPI to list IDPs, show login URLs to the user for logging in via all IDPs. It also uses the SPAPI to constructs links for federation termination.

Sources

- **web/index.jsp**: Contains almost all the source code for this sample. This is the index page and the login page. It also has the logout and federation termination functionality.
- **web/activate.jsp**: This is the activation page as in the SP-Activation sample. It displays the user's activation profile and tries to associate it with a local login or assign a new local user ID to the activated user.

Running the SP-Sample Sample

To run the `sp-sample` sample, perform the following steps:

1 Deploy the `dist/sp-sample.war` file to your J2EE server.

2 Navigate to the `index.jsp` page.

The `index.jsp` page displays a login prompt and links for logging in through configured IDPs.

If you do not see any links to login through the IDPs, you have not configured Authority sites in your circle-of-trust. Do the following to configure Authority sites:

a Click on the **login via IDP** link to navigate to the IDP.

b Login as the user.

  You are redirected back to the SP.

If the user logged in at the IDP, which is not federated with the SP, you will be navigated to the Activation page.

3 On the Activation page, you can do one of the following:

- Associate the user with a local account.
- Assign a new user ID to that user at the SP site.

You are then navigated to the Index page.

4 On the Index page, you can initiate a global logout or terminate the user's federation with the IDP that the user is logged in from.

IDP-SampleAuthnPlugin

The `idp-SampleAuthnPlugin` sample demonstrates how to use the `IDPAuthnPlugin` Interface.

The sample defines the class `AuthnPlugin` which implements the interface `IDPAuthnPlugin`. As defined in the API documentation, the `IDPAuthnPlugin` class is constructed by Select Federation and is specified in the `tfsconfig.properties` file by the following line:

```
# IDP Authentication Plugin
idpAuthnPlugin=AuthnPlugin
```

Alternatively, the plugin can be configured with both a class and a path to a `jar` file:

```
# IDP Authentication Plugin
idpAuthnPlugin=myauthnplugin
myauthnplugin.class=AuthnPlugin
myauthnplugin.jar=<cd-base-directory>/dist/authnplugin.jar
```

When Select Federation requires user authentication, it instantiates the implementation of the IDPAuthnPlugin Interface using the following constructor where the `conf` object points to the `tfsconfig.properties` file:

```
IDPAuthnPlugin (Config conf)
```

Select Federation then calls the `authenticateUser` function, passing in the request and response objects, the provider ID of the requesting application (SP) site and whether the authentication requested is a *passive* one, which can be done without user interaction. It also passes in the authentication context class desired. The sample shows a simple implementation of this function where it does not support a passive authentication at all, and shows a simple login page to the user, ignores the password and provides the user ID back to Select Federation.

The sample implements a trivial logout method, but in a real integration this would initiate a logout from the local Identity Management System.

Running the IDP-SampleAuthnPlugin Sample

➤ This sample must be run on the same J2EE server as Select Federation.

To run the `idp-SampleAuthnPlugin` sample, perform the following steps:

1 Open the `tfsconfig.properties` file and comment the line for the default idpA`uthnPlugin` setting (add # in the beginning):

```
# idpAuthnPlugin=com.trustgenix.tfsIDP.util.IDPAuthnPlugin_Dir
```

2 Add the following lines in the `tfsconfig.properties` file on the IDP site.

```
# IDP Authentication Plugin
idpAuthnPlugin=myauthnplugin
myauthnplugin.class=AuthnPlugin
myauthnplugin.jar=<cd-base-directory>/dist/authnplugin.jar
```

3 Restart the service.

IDP-SampleDirPlugin

The `idp-SampleDirPlugin` sample demonstrates how a Directory plugin can be written. The sample provides the sources for equivalent functionality as the `DirPlugin` file, which ships as a part of Select Federation.

The sample defines the class `sampleDirPlugin` which implements the Interface `DirPlugin`. As defined in the API documentation, the `DirPlugin` class is constructed by Select Federation and is specified in the `tfsconfig.properties` file by the following line:

```
# Directory Plugin
# dirPlugin=com.company.tfsIDP.util.DirPlugin_LDAP
dirPlugin=SampleDirPlugin
SampleDirPlugin.filePath=conf/SampleDirPlugin.properties
```

Alternatively, the plugin can be configured with both a class and a path to a jar file:

```
# Directory Plugin
dirPlugin=mydirplugin
mydirplugin.class=SampleDirPlugin
mydirplugin.jar=/path/to/dirplugin.jar
```

When Select Federation requires user authentication, it instantiates IDPDirPlugin for each profile query call. In addition, the `verifyPassword` call may be made by an alternative source (such as an implementation of the IDP`AuthnPlugin`, so the IDP`DirPlugin` may be instantiated multiple times).

Running the IDP-SampleDirPlugin Sample

➤ This sample must be run on the same J2EE server as Select Federation.

To run the `idp-SampleDirPlugin` sample, perform the following steps:

1 Copy the `sampleDirPlugin\sampleDirPlugin.properties` file to the `conf` directory on the IDP site.

2 Open the `tfsconfig.properties` file and comment the line for the default `dirPlugin` setting (add # in the beginning):

```
# dirPlugin=com.company.tfsIDP.util.DirPlugin_LDAP
```

3 Add the following lines in the `tfsconfig.properties` file on the IDP site.

```
dirPlugin=mydirplugin
mydirplugin.class=SampleDirPlugin
mydirplugin.jar=<cd-base-directory>/dist/idpplugins.jar\
```

4 Restart the service.

IDP-AuthDir

The `idp-AuthDir` sample shows the effective use of the `IDPAuthnPlugin_Dir` which is an Authentication plugin shipped with the Select Federation binary distribution. The `IDPAuthnPlugin_Dir` obtains the username and password from the login page and verifies the password by invoking the Directory Plugin's `verifyPassword` method.

You will require a J2EE application server (which can be used to host the sample as well as Select Federation).

This sample requires the Authentication plugin to be set to the `IDPAuthnPlugin_Dir` and any Directory Plugin to be specified.

Sources

- **`web/dirlogin.jsp`**: This is the login page for users requiring authentication against the directory. This page is redirected to by the IDP Single Sign-On Service in Select Federation. This page shows the user a login form and posts the results back to the IDP Single Sign On Service.

Building the Sample

The command to build the sample is:

```
ant clean package
```

Running the IDP-AuthDir Sample

To run the `idp-AuthDir` sample, perform the following steps:

1 Edit the `tfsconfig.properties` file to add the following line.

```
idpAuthnPlugin =com.trustgenix.tfsIDP.util.IDPAuthnPlugin_Dir
```

This line configures the `IDPAuthnPlugin_Dir` Authentication plugin for authenticating users against a directory.

2 Edit the `tfsconfig.properties` to configure the `loginURL` property.

This property must be set to the URL corresponding to `dirlogin.jsp` of the sample.

You must restart Select Federation after you edit the `tfsconfig.properties` file.

3 Deploy the `idp-authdir.war` to the J2EE application server.

4 Go to a properly configured SP and attempt a federation.

You are redirected to the IDP's login page (`dirlogin.jsp`).

5 Enter a username and password that can be verified against the directory to login in at the IDP.

If the entered combination is incorrect, you will see the login page again.

A simple way of testing this sample is to use the File plugin that is created as part of the Select Federation installation. The file is called `users.properties`, and can be found under `$SF_HOME/properties`.

SP-Activation

This sample contains code that uses the built-in access filter and the SPAPI to provide activation and application integration. The Access Filter is configured through servlet initialization parameters specified in the file `web/WEB-INF/web.xml`.

Sources

- **`web/index.jsp`**: Is the main page which is shown after the user is logged in and activated.
- **`web/activate.jsp`**: Is the activation page, configured in `web/WEB-INF/web.xml`. This page is executed when a user is logged in via an IDP, but does not have a local federation at the SP.
- **`web/login.jsp`**: Is the local login page. The user may login locally or follow a link on this page to login via an IDP.
- **`web/logout.jsp`**: The page that used to globally and locally logout the user.

Running the SP-Activation Sample

Deploy `sp-activation.war` on your J2EE server. (It does not have to be the same one as Select Federation, only has to share the federation repository with it.) After deploying `sp-activation.war` navigate to the `index.jsp` page. The access filter will kick-in and show you the login page (`login.jsp`). On this page, you can choose to login locally or login via a configured IDP. If you don't see a list of links from the configured IDPs, your circle-of-trust has not been configured to have any IDPs.

If you click to login through an IDP and login at the IDP as a user who has never been federated with the site at which you have deployed the SP-Activation sample, you will be navigated to the Activation page (`activate.jsp`).

On the Activation page, you will see the activation profile parameters of the user. If you do not see the profile parameters, the IDP probably does not have the ID-EP or ID-PP service properly configured or running. On the Activation page, you can associate the user (who has been logged in through the IDP) with an existing local account (enter any user ID here) or the sample can generate a unique user ID (click the **Continue** button).

IDP-Portal

This sample demonstrates IDP initiated Single-Sign-On (SSO). This is a typical scenario when using the SAML protocol. This sample requires that you do the following:

- Configure the Home page URL parameter while specifying the SPs in the circle-of-trust. This sample dynamically lists all SPs that are in the circle-of-trust and have the Home page URL specified.

- Deploy the `idp-sample` at the IDP. This is because this sample acts like a local SP and requires users to be authenticated using the local IDP, which is serviced by `idp-sample`. As in the previous sample, SP-Activation, you need to set the loginURL to the `idp-sample/login.jsp`.
- Set the `defaultIDP` property in the `idpconfig.properties` file to the same Select Federation `providerId`. If you did not modify the `defaultIDP` property in the installation script, this will be automatically true.

Sources

This sample has only one source file `index.jsp`.

Running the IDP-Portal Sample

Compile and deploy the `idp-portal.war` to an appropriately-configured application server. Then open a browser and navigate to the `/idp-portal` URL of that application server.

IDP-Intro

This sample describes how to use the Liberty introduction protocol to find the IDP to which a user belongs. This sample allows the user to click a link to set the introduction cookie.

This sample works with the SP-Intro sample that allows the user to click a link to read the introduction cookie. This sample can only be used when Select Federation is using the artifact profile.

Sources

- **`web/index.jsp`**: This page is seen by the user after logging in. This page has a link which can be used to set the Introduction cookie. If the user is not logged in this page displays a form to login the user.
- `web/login.jsp`: This is the page used by Select Federation to login the user when coming from an SP requiring user authentication.

Running the IDP-Intro Sample

The `tfs-intro WAR` in Select Federation should be deployed on an application server that is in the cookie domain set below:

Configuration properties in `tfsconfig.properties`:

```
Set the domain of the cookie by setting the
following property (note that the value begins
with a ".")
```

```
cookieDomain=.commondomain.com
```

Unless SSL is turned on, **do not** set the following property. This property or cookies may not be secure. (Only set this property when SSL is turned on, or for debugging purposes.)

```
cookieSecure=false
```

Configuration properties in `idpapiconfig.properties`:

```
Set the URL for the common domain cookie writer:

cookieWriterServiceURL=http://idp.commondomain.com/tfs-intro/
CookieWriterService

Set whether or not to use the cookie writer:

useSSOCookieWriter=1
```

SP-Intro

This sample builds on `sp-sample` and demonstrates the use of the Liberty ID-FF Introduction cookies. This sample can read Liberty-compliant Introduction cookies and login users based on such cookies. An IDP-side example that sets such cookies is IDP-Intro. This sample works only when using the artifact profile.

To use this sample, you will need the following:

- J2EE application server (which can be used to host the sample as well as Select Federation).
- To install and configure your SP profile with at least one IDP.

Sources

- **`web/index.jsp`:** Contains all the source code for this sample.

Building the Sample

The command to build the sample is:

```
ant package
```

Extra Configuration Parameters

The following configuration files require additional configuration parameters to run this sample:

On the SP side: –`tfsconfig.properties`

URL prefixes which are allowed to be returned to/from the cookie reader. This is required to prevent an attacker from reading your intro cookies from any arbitrary site.

```
cookieReaderReturnPrefixes=http://<sp.commondomain>
```

On the SP side: `spapiconfig.properties`

URL for the common domain cookie reader.

```
cookieReaderServiceURL=http://sp.commondomain.com/tfs-intro/
CookieReaderService
```

Running the SP-Intro Sample

After deploying the `sp-intro.war` file on your J2EE application server, load the `index.jsp` page in a web browser.

Establish a federation with an IDP

1. Enter a username and password to log in to the local SP account that you wish to federate with an IDP. Any username and password will be accepted by the sample code.
2. Once logged in, click **Federate** for the configured IDP that you wish to federate the SP account with.

   You are redirected to the IDP.
3. Log into the account at the IDP that you wish to federate your SP account with.

   You should be returned to the SP and logged in.

Login using Introduction Cookies

1. In a new browser window, browse to the IDP-Intro sample to set the Introduction cookie.
2. Login locally at an IDP, then navigate to the `sp-intro/index.jsp` page.
3. Click **Read Introduction Cookie**.
4. In the list of IDPs that are shown, click on an IDP to login using that IDP.

Login using your federation

1. Ensure that you are logged out from the application.

   Click **logout** if needed.
2. Click on the link to login through the IDP that you federated your account with.
3. Log into the account at the IDP that you used to establish the federation.

   You should be returned to the SP and logged into your local account.

Terminating your federation

1. Log into the local SP account, either using the local login form or a federated login.
2. Click **Terminate Federation** for the IDP federation that you wish to terminate.

IDP-Proxy

This sample demonstrates the use of the proxy function for authentication at an SP site using the “Home IDP” of a user that is not trusted by the SP. The SP trusts a “proxy IDP” that in turn trusts the home IDP. The proxy IDP acts as an SP for the home IDP and as an IDP for the SP.

This sample is used on the proxy IDP, and works with `sp-sample` on the SP side and `idp-sample` on the home IDP side.

Sources

- **`web/login.jsp:`** Provides a way for the user to login using the home IDP or login locally.
- **`web/index.jsp:`** Provides a way for the user to initiate a logout from the proxy IDP.

- **`web/logout.jsp`:** Provides a way for the IDP to continue an SP-initiated logout and to logout the user from the home IDP.

Prerequisites

The following software is required to run this sample:

- Ant 1.5
- JDK 1.4.2
- J2EE 1.3.1

Following are other requirements:

- You must have a Select Federation IDP configured with at least one SP and one other IDP.
- Select Federation is configured to do an http-based logout and not a SOAP-based logout.

Building the Sample

The command to build the sample is:

```
ant package
```

Running the IDP-Proxy Sample

To run the idp-proxy sample, perform the following steps:

1. Configure two IDPs (IDP1 and IDP2) and one SP (SP1).
2. Modify the following `conf/tfsconfig.properties` file parameters in the two IDPs, so that the parameters have the following values:

   ```
   idpSingleLogoutProtocolProfiles=http://projectliberty.org/profiles/
   slo-sp-http
   ```

   ```
   spSingleLogoutProtocolProfiles=http://projectliberty.org/profiles/
   slo-idp-http
   ```

   (Remove the `slo-sp-soap` and `sl-idp-soap` profiles.)
3. Set the login URL of the IDP2 to be the following:

   `loginURL=${SITENAME}/idp-proxy/login.jsp`
4. Download the metadata for all three sites as the Liberty 1.2 combined IDP+SP metadata.
5. Upload the IDP2 metadata into IDP1 as an "Application (SP)" site.
6. Upload the IDP1 metadata into IDP2 as an "Authority (IDP)" site.
7. Upload the IDP2 metadata into SP1 as an "Authority (IDP)" site.
8. Upload the SP1 metadata into IDP2 as an "Application (SP)" site.
9. Deploy the sample `idp-sample` on IDP1.
10. Deploy the sample `idp-proxy` on IDP2.
11. Deploy the sample `sp-sample` on SP1.
12. Navigate to the URL for the `sp-sample` and click **Login via <proxy IDP>**.

    A login page opens.

13 Click **Login via <home IDP>**.

The Home IDP login page opens.

14 Enter a user name (no password required).

The Activation or logged in page of the `sp-sample` opens.

15 Click **logout** and you are logged out of both IDPs.

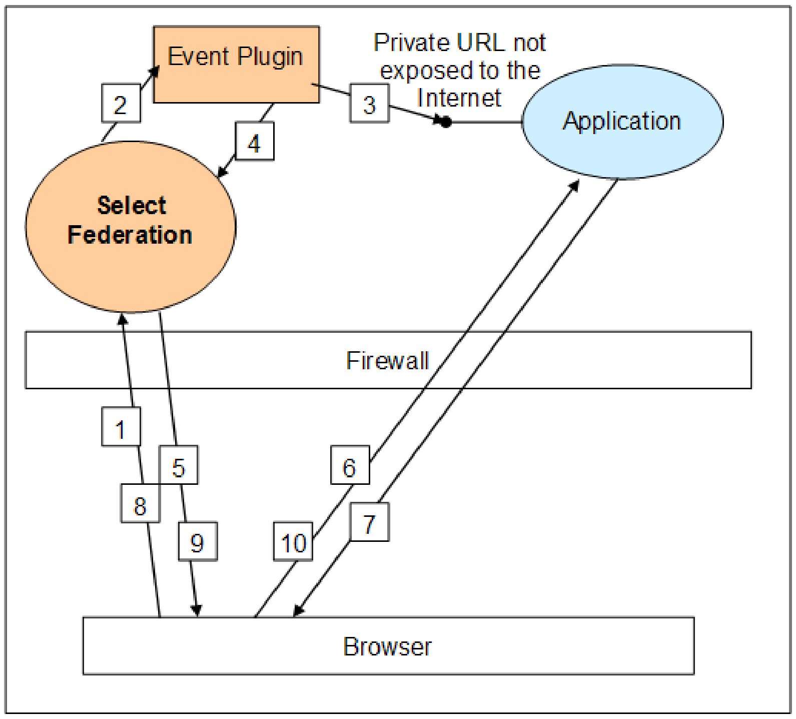
sp-SampleEventPlugin

Although the Event Plugin sample is just like the other samples provided on the CD, it is special in that it can be used in a production environment without modification to provide unique notification features for integration with non-Java applications.

The Event Plugin sample is located in the directory: `sp-SampleEventPlugin`.

This sample allows applications to be either called or redirected to when a user logs in, logs out or the user's federation is terminated. The URL Call mechanism may be used to provide a secure way of notifying the application about the details of the user. This works as described in the following diagram:

Figure 3 Event Plugin Sample Process



The sequence of events is as follows:

1 The user is redirected to Select Federation in the process of a federated login by a user.

2 Select Federation calls the Sample Event Plugin.

The Event Plugin may be configured to do a URL Call or to redirect.

3 (Optional) If configured to do a URL Call, the Event plugin calls out to the configured URL in the application. For security reasons, this URL should not be available to the Internet because data obtained through HTTP Headers or URL parameters to this URL is trusted information.

4 If configured to do a URL Call, the Event plugin writes any cookies sent to it by the Private URL back to the browser. If the event plugin is configured to do a redirect, it writes the redirect to the browser via Select Federation.

5 Select Federation passes the response from the Event plugin to the browser.

6 Steps 6 – 8 are done only if the Event plugin is configured to a redirect and not a URL call. If configured to do a redirect, the browser is redirected to the login notification URL configured in the event plugin due to the redirect written to the browser in Step 4.

7 The application obtains information about the user by calling the Select Federation API (not shown) and redirects back to Select Federation. (It may also set a session cookie in the browser.)

8 Select Federation receives the redirect back from the application.

9 Select Federation redirects the user to the Target URL where the user intended to go when Select Federation first received the user from the federated site. If the request originated at the SP site, then the user is taken back to the return URL specified in the SPAPI call that initiated the federated login. If the request originated at the IDP site and it doesn't specify the target URL, the user is redirected to the application URL specified in the Select Federation management console.

10 The application receives control at the target URL.

Advantages of the Call URL Method

The Call URL method, in which Select Federation and the Event plugin act as an HTTP client to the Application's private URL, is useful when the Application is not a Java Application or for some other reason cannot call the Select Federation APIs. This way, the user information can be passed in either URL parameters or HTTP headers. Once the Application knows about the user, it can set a cookie in the response that will be set in the user's browser, so when the user reaches the Target URL, the Application will be able to recognize the cookie and know the identity of the user. It also avoids two additional redirects to the Application for event notification.

Sources

- **`src/SampleEventPlugin.java`**: Instantiates the SPEventPlugin class and redirects to or calls as an HTTP client any URLs specified.
- **`web/callLogin.jsp`**: This is a sample Responder page that can be called by the Sample plugin. It logs the authenticated user ID and session ID to the log file.
- **`web/redLogin.jsp`**: This is a sample Responder page that can be redirected to by the Sample plugin. It displays the user ID and any profile information on the browser screen.

Running the sp-SampleEventPlugin Sample

This sample must be run on the same J2EE server as Select Federation.

To run the sp-SampleEventPlugin sample, perform the following steps::

It is recommended that you run this sample in standalone mode only.

1. Open the `tfsconfig.properties` file on the SP site.
2. Add the following lines.

   ```
   spEventPlugin=myeventplugin
   myeventplugin.class=SampleEventPlugin
   myeventplugin.jar=<cd-base-directory>/dist/spplugins.jar
   ```

3. Deploy the file `dist/EventResponder.war` to your application server.
4. Restart the service.

Configuration

The Event plugin requires the following variables (if any of these are not set, it either does not execute that part of the functionality or provides reasonable defaults). All of the parameters mentioned below are always set as:

```
SampleEventPlugin.<param-name>=<value>
```

Table 1 sp-EventPlugin Parameters

Parameter Name	Description
`redirectToURLs`	This variable controls whether the Event plugin redirects to URLs or acts as an HTTP client to call the URLs. Set this value to zero (0) for calling the URLs.
`spLoginURL`	The URL invoked (called or redirected to) when a federated user logs in
`spLogoutURL`	The URL invoked (called or redirected to) when a federated user is logged out.
`spActivateURL`	The URL invoked when a new user visits the SP for the first time.
`spDeactivateURL`	The URL invoked when a user has been de-federated with a particular IDP.

Table 1 sp-EventPlugin Parameters

Parameter Name	Description
`sessionHeaderName`	When being called, URLs are passed the federated session ID in this HTTP header. The default value is IB_SID.
`uidHeaderName`	When being called, URLs are passed the federated user ID in this HTTP header. The default value is IB_UID.
`cookieDomain`	After calling the `spLoginURL`, the sample event plugin sets all cookies set by `spLoginURL` in the response to the browser. If this parameter is set, those cookies are set under this domain.

6 Troubleshooting

To troubleshoot effectively, be sure to configure a log file location, and enable debug logging. If your issue is not covered in this appendix, be sure your log file is available when reporting the issue to Support (see Support on page 4 for information).

Error Messages

Error Message

Got unexpected Exception...please try again.
"Exception: com.trustgenix.tfs.TFSException: SSO Failed, probably because of a missing cookie"

Solution

The cookie names specified in the Select Federation SP instance configuration file and the filter configuration should be the same. Do the following:

1. Open the `tfsconfig.properties` file of the SP instance and search for `filterSupport.cookieName`.
2. Confirm that the value of this attribute is the same as the cookie name being set as part of the Select Federation filter configuration.

➤ A good way to debug issues related to cookies is to change the browser setting so as to **prompt** before cookies are set.

Error Message

Missing IDP

Solution

To test the SF filter, you need to have configured an IDP that has been set up to validate users against an Access Management system (such as Select Access) or a Directory Server (such as Active Directory).

Error Message

Java Index OutofBounds exception when accessing protected resource

Solution

This can happen if you tried to access a URL without specifying the protected resource. For the filter to function correctly, the resource that is being accessed should have three components:

- protocol

- server name
- resource name

Therefore, when accessing the protected resource, be sure to specify the full URL containing all three components. For example:

http://myserver.com/myprotectedresource

Error Message

http://localhost/SFFilterConfigure.html is "not found" [IIS filter only]

This could happen if the Select Federation filter is not loaded.

Solution

- Make sure that the filter has been added to the web service extension list, and that the status is set to **Allowed**.
- The machine should have been rebooted after modifying the system path.

Glossary

Access Control

The authorization policies and conditions that regulate identity access to resources with a goal towards preventing unauthorized use or use in an unauthorized manner.

ADFS (WS-Federation 1.0)

Active Directory Federation Services (ADFS) is a feature of Microsoft Windows 2003 Server R2. ADFS allows a federation with Active Directory-based users, by using the WS-Federation 1.0 protocol.

Administrator

An identity with full permission to manage Select Federation.

Application Helper

Select Federation component that helps you configure URLs in your application for seamless navigation to the Service Provider (SAML Consumer) sites or for authentication through the Identity Provider (SAML Producer) sites.

Application Site Role

An application site (also called a SAML Consumer or Service Provider (SP) Site), which is a Trusted Partner site that participates in a federation to provide a service or application to common users and relies on an authority site to provide authoritative user authentication and other information. For example, in a federation of an extranet with partners' corporate portals, the site hosting the extranet is the application site.

ASP

Microsoft Active Server Pages log users in by invoking the IDP-FSS over a secure channel.

Attribute

One or more characteristics that are part of an identity profile. Attributes are name/value pairs with a type that is assigned a value. For example, an attribute called "Department" may be assigned the values of, "IT", "Sales", or "Support". These attributes are interpreted and assigned appropriately to profiles in different applications (LDAP-compliant directories, databases, SAPs, and so on) based on the mapping rules defined for that application.

Authentication

The act of verifying the credentials of an identity and matching them with an identity profile. The evaluation of credentials ensures that the identity is truly who or what they claim to be.

Authority Site Role

An authority site (also called a SAML Producer or Identity Provider (IDP) Site), which is a Trusted Partner site that participates in a federation to authenticate users and provide other authoritative user information to other sites. For example, in a federation of an extranet with partners' corporate portals, the portals act as the authority site.

Authorization

The process of defining and enforcing the entitlements of an identity. Authentication is a prerequisite for authorization. See Access Control and Authentication.

CA

Certificate Authority

CSR

Certificate Service Request

Delegated Administrator

An identity that has been added by the root administrator. The delegated administrator can perform all functions that the root administrator performs except admin-related functions such as add and remove admins and change admin passwords. When Select Federation is running in Standalone mode, the delegated administrator also cannot view the Admin Audit log. But when Select Federation is integrated with Select Access, then the delegated administrator can view the Admin Audit log. See Root Administrator.

DS

Discover Service

DST

(Data Services Template) DST-based services such as the Personal Profile service (ID-PP) and the Employee Profile service (ID-EP).

Federation

The combination of business and technology practices to enable identities to span systems, networks and domains in a secure and trustworthy fashion. This is analogous to how passports are used to assert our identity as we travel between countries.

ID-WSF

Liberty Identity Web Services Framework security mechanism.

IDP

An Identity Provider or IDP is an organization or web site that asserts the identity of users to the Service Providers or SPs in a federated network. The assertion of the user identity is done using standard protocols such as SAML and Liberty.

IDP-FSS

IDP filter-support service, which is a servlet component of the Integrated Windows Authentication (IWA). The IDP-FSS enables a trusted program to add a Windows-authenticated user ID into an IDP session.

IIS

The Internet Information Server (IIS) is the web server that is bundled with Windows 2003 Server.

Integrated Windows Authentication (IWA)

Allows Select Federation to leverage a user's Windows logon credentials to seamlessly authenticate the user and transfer the user to a Trusted Federation Partner site.

Keystore

A keystore is a database of keys. The private keys are associated with a certificate chain, which authenticates the corresponding public key. The keystore also contains certificates from trusted entities. By generating the keystore, you add another layer of security to the data that is exchanged in the Select Federation system.

LDAP (Lightweight Directory Access Protocol)

A set of open protocols for accessing information directories. LDAP can make the physical network topology and protocols transparent so that a network identity can access any resource without knowing where or how it is physically connected.

LECP

Liberty Enabled Client/Proxy Service.

MMC

Microsoft Management Console, used to set up server authentication and to import the `pkcs` / `pfx` format file into your local store on the IIS machine.

NTLM

NT LAN Manager [web definition: is a challenge/response form of authentication that was the default network authentication protocol in Windows NT 4.0.]

Protected URLs

Protected URLs require users to be authenticated to allow access to these URLs. If a user is not authenticated, the filter redirects the user to Select Federation for authentication. The Select Federation installation may authenticate the user locally or initiate federated login at another Authority (IDP).

Passive URLs

Passive URLs are for resources where users' personalized content is not critical for the application. Users are allowed to access these URLs even though they cannot be authenticated without being prompted. However, if the user is already logged in at the IDP, has a federation session with Select Federation, or can be authenticated without being prompted, the user's identity and attribute information is presented in the federation session to the application.

Presence Service

A service that informs the WSC if a user is online, available, and so on.

Root Administrator

The “super user” administrator who has complete entitlement to all functionality in the Select Federation Administration Console. The root administrator’s login is always “admin”. Only the root administrator can add and remove delegated administrators and change administrators’ passwords. See Delegated Administrator.

SOAP

Simple Object Access Protocol is a fundamental web services standard for XML-based communication between web service providers and consumers.

SP

A Service Provider (SP) is an application that allows authenticated access based on an authentication performed by an IDP using a federated identity protocol such as Liberty or SAML.

SSL

Secure Sockets Layer handshake protocol, which supports server and client authentication.

SSO

Single Sign-On session/authentication process that permits a user to enter one set of credentials (name and password) to access multiple applications. A Web SSO is a specialized SSO system for web applications.

SAML

Security Assertion Markup Language protocol.

Unprotected URLs

Unprotected URLs allow users access to these URLs without being authenticated. Typically, special URLs such as the login URL and logout URL are unprotected URLs.

WSC

A Web Service Consumer (WSC) is an application that uses web services. It may not be a web service in itself, but uses XML and typically SOAP-based communication with a web service to perform some of its functions.

WSP

A Web Service Provider (WSP) is a web service application that services requests it receives based on XML and typically SOAP-based communication.

Index

invent

Java™
COMPATIBLE