HP OpenView Select Federation

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

Software Version: 6.5

Web Application Developer's Guide

Document Release Date: April 2006
Software Release Date: April 2006

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

HP OpenView Select Federation includes software developed by third parties. The software in Select Federation includes:

- Software developed by Trustgenix, Inc. Copyright © Trustgenix, Inc. 2002-2005. All rights reserved.
- Apache Derby, Apache Xalan Library, Apache Xerces Library, and Apache XML Dsig Library.
- Software developed by the University Corporation for Advanced Internet Development <http://www.ucaid.edu>Internet2 Project.

Trademark Notices

- Linux is a U.S. registered trademark of Linus Torvalds.
- Microsoft®, Windows®, and Windows NT® are U.S. registered trademarks of Microsoft Corporation.
- Oracle® is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation.
- UNIX® is a registered trademark of The OpenGroup.

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

Please visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter 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 log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://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.

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.

Audience

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

Prerequisites

This guide assumes a working knowledge of:

- Identity Management
- Federated Identity
- HP OpenView Select Federation Architecture Guide
- The basics of popular web application development technologies such as HTTP, HTML, Java Server Pages and CGI scripts

System Requirements

HP OpenView 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 Chapter 3 of the *HP OpenView Select Federation Configuration and Administration Guide* for the specific hardware and software system requirements.

Chapters Summary

The table that follows provides an overview of this guide's contents.

Table 1 Chapters Summary

Chapter	Description
Chapter 1, Introducing the HP OpenView Select Federation Web Application Developer's Guide	This chapter provides a brief overview of the content of this Developer's Guide.
Chapter 2, Overview	This chapter introduces federation functionality and how to enable applications and authorities for a federation.
Chapter 3, Creating a Test Setup	This chapter explains how to create a test environment which can be used to deploy and run the samples.
Chapter 4, Using Filters to Protect Web Applications	This chapter describes the HP OpenView Select Federation filters.
Chapter 5, Select Federation API Overview	This chapter provides a brief overview of the Select Federation APIs and how to access the Select Federation API documentation.
Chapter 6, Activation and Auto Enrollment	This chapter describes how new users can be activated and auto-enrolled at a Service Provider or a SAML consumer, when they visit the site for the first time.
Chapter 7, API Samples	This chapter provides descriptions of several API samples to help developers understand the capabilities of Select Federation.
Appendix A, How Filters Work	This appendix describes how filters work to protect web applications.

Table 1 Chapters Summary

Chapter	Description
Appendix B, Supported Apache Versions	This appendix provides version information for the Apache filter run-time dependencies on Linux platforms.
Appendix C, Configuring Server and Client Authentication	This appendix provides a scenario to set up the Apache filter to configure server and client authentication.
Appendix D, Troubleshooting	This appendix provides troubleshooting methods for problems you may run into.

2 Overview

This chapter provides an overview of the functionality that can or should be added to enable applications for federation. This chapter also introduces the various ways offered by HP OpenView Select Federation and its SDK to create this functionality.

This chapter includes the following topics:

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 (perfectly possible in a federated deployment!), or does not have local user identifiers that were established before 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 3, 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, not yet linked accounts, it can choose to do so now, a process that is called *activation* (see Chapter 6, 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

Simply 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 as well as an authority can ask the other party to terminate a federation. This is not often available to end users as an option because they rarely need it and it is difficult to explain. The `sf-demo` has links for this to enable testers to remove and re-establish federations.

Now that the federation concepts are somewhat familiar, we can discuss what it means to enable applications and authorities.

Enabling Applications

An application may wish for 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.

HP OpenView 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 HP OpenView 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 often 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 Chapter 4, Using Filters to Protect Web Applications for detailed information.

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 50 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 5, 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 Chapter 11 of the *HP OpenView Select Federation Configuration and Administration Guide* for more information.

Authority Relies on HP OpenView 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 helper applications can be used to construct Single-Sign-On (SSO) URLs to the application partners. See the “Using the Application Helper” section in Chapter 8 of 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 58.

Alternatively, the authority can make use of a custom-made *Authentication plugin* (see IDP Authentication Plugin Interface on page 50 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 58 for a description of using the IDPAPI. For an overview of the IDPAPI, see IDP API on page 50.

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 62 for a description of using the `IDPAuthnPlugin_Dir` plugin. For an overview of the IDPAPI, see IDP API on page 50.

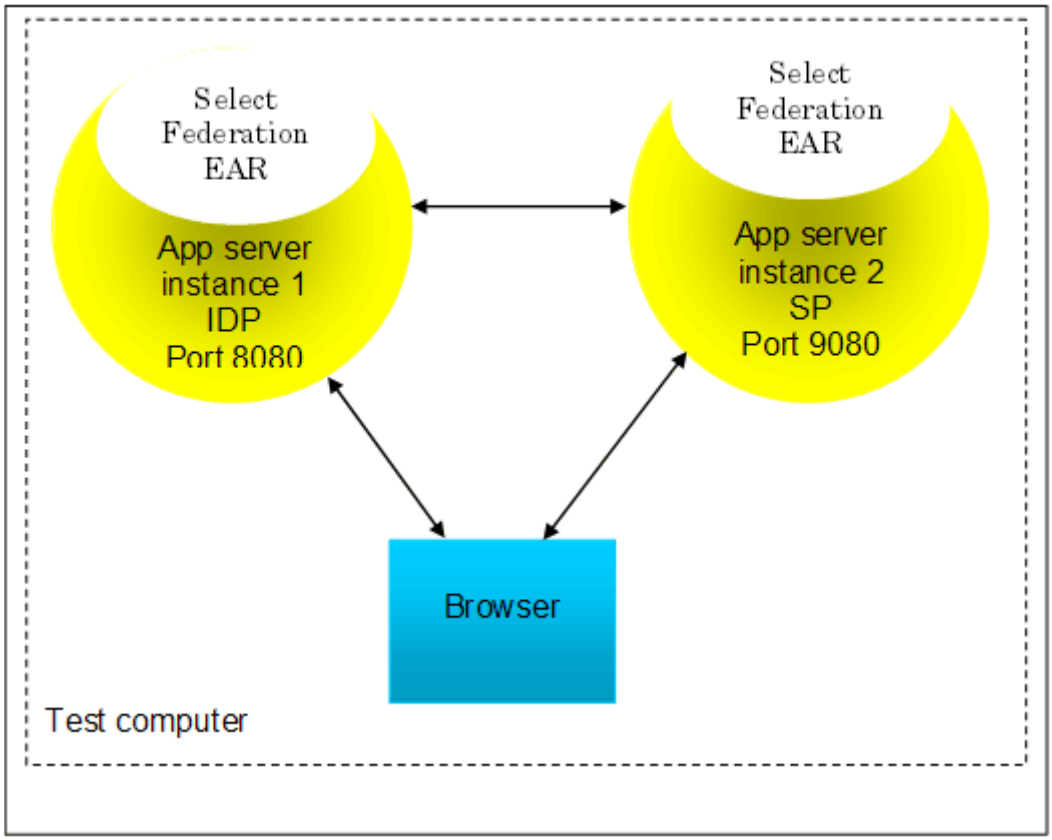
Other user-related services can be exposed as standards-based identity web services. Examples of such services include: geolocation, calendar, and so on. See the *HP OpenView Select Federation Web Services Developer's Guide* for more information.

3 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



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**.

If you configured `https` as the preferred protocol during the Select Federation installation, be sure that the Certificate Authority (CA) that issued the server SSL certificate is included in the Java trust list **at the partner site**. For example, if your installation is a Service Provider that was issued a self-signed SSL certificate by Select Federation during installation, that certificate must be imported in the java `cacerts` file of the Identity Provider. Look for a certificate called **tomcat.cer** under `<sf_home>/conf`, and import it into the IDP's trust store. You could use a simple java utility like `keytool` to install the CA's certificate in your Java `cacerts` file. A sample `import` command will look somewhat like this:

```
% keytool -import -trustcacerts -alias MyCA -file <CA certificate file>
-keystore <path to your Java installation>/lib/security/cacerts -storepass
<default value is "changeit">
```

The same will hold if the SP side SSL certificate is issued by a third party CA. In that case, the only change would be to import the third party CA certificate instead of the SP self-signed certificate. Note that the above process needs to be repeated at the SP if the IDP is also deployed over SSL.

The location of the java `cacerts` file will vary depending on the application server you chose during installation. The default paths are as follows:

WebLogic: `BEA_HOME/jdk142_05/jre/lib/security/cacerts`

WebSphere: `IBMWS_HOME/java/jre/lib/security/cacerts`

Built-in application server: `SF_HOME/_jvm/lib/security/cacerts`

Follow the steps in Chapters 4, 5, and 6 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 (see Chapter 4, Using Filters to Protect Web Applications) require attributes of authenticated users to be available at the SP application. For a detailed description of Attribute policy configuration, see the “Editing Partner Settings” chapter 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.

4 Using Filters to Protect Web Applications

This chapter describes the HP OpenView Select Federation filters in the following topics:

Overview of Filters

A filter is a software component that changes the default behavior of a web server and determines how the HTTP requests and responses are handled. As a component at the Service Provider (SP) site installation of Select Federation (SF), the filter provides the following functionality:

- Validates user authentication or Single-Sign-On (SSO) before a web server serves an HTTP request for the following classes of URLs:
  - **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.
  - **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.

  The filter validates user authentication or SSO by looking for a special cookie (default cookie name `SFSession`) in the cookie header of the request. If the cookie header contains the special cookie, then the filter assumes that the user is authenticated. Otherwise, the filter redirects the user to the login page.
- Allows the user-specific information (such as profile and login information) to be available to the web applications by setting the request headers. See Appendix A, How Filters Work for more details.

Select Federation provides the following filters to change the default behavior of non-Java web servers and to provide the functionality just described:

- **IIS Filter** — Uses the IIS Internet Server Application Programming Interface (ISAPI), which is provided by Microsoft for IIS web servers running on Windows.
- **Apache Filter** — Uses the Apache API, which is provided by Apache 2.0 for Apache web server(s) running on Windows or UNIX platforms.

The Select Federation Apache 2.0 module binaries are compatible with Apache 2.0.41+ versions that have a Module Magic Number that starts with 20020903. See Apache Filter on page 39 for more information.

However, the Select Federation Apache 2.0 module binaries are not compatible with Apache 1.3.

In some cases (such as during evaluation) it is not practical to install such a filter in the web server. Therefore, a PHP script is included on the Select Federation CD, which does the same work as the filters. This script may be helpful in better understanding the principles involved. See the `<cd-base-directory>\filters\custom\` directory for the PHP script.

Filter-Support for Non-Java Web Servers

The Select Federation SDK includes a dedicated Java web application that is used to integrate with the filters provided for the non-Java web servers mentioned above (or with web servers that cannot contact the Select Federation databases which are normally kept behind a firewall). This application is intended to be deployed on the same machine as the (SP) install of Select Federation.

How to Configure Filter-Support

Perform the following steps to configure Filter-Support:

1. Deploy the `<cd-base-directory>\filters\support\tfsfs.jar` file on your Application Server where you have installed your SP Select Federation instance.

   The `tfsfs.jar` file contains the following files:

   - `com.hp.ov.selectfederation.filters.support.FilterSupportPlugin`: This is an implementation of the EventPlugin that will take care of setting the cookies on your behalf and can be configured for use in the `tfsconfig.properties` file of your SP installation. For a description of the EventPlugin interface, see SP Event Plugin Interface on page 50. For a sample of applying the EventPlugin, see sp-SampleEventPlugin on page 67.
   - `com.hp.ov.selectfederation.filters.support.FSService`: This is a servlet that communicates between the filters and the Select Federation database. This servlet retrieves data from the database based on the filter configuration. When configuring the filters, you need to reference this service as a URL. For instructions on configuring the FSService, see How to Configure the IIS Filter on page 30 and How to Configure the Apache Filter on page 42.

2. Deploy the `<cd-base-directory>\filters\support\tfsspwscapi.jar` file on your Application Server where you have installed your SP Select Federation instance.
3. Deploy the `<cd-base-directory>\filters\support\tfsspapi.jar` file on your Application Server where you have installed your SP Select Federation instance. If you have an older version of the `tfsspapi.jar` file, be sure to delete that version and use the newer one provided on the Select Federation CD.

4 Edit the configuration of the SP installation.

a Open the <*sp-installation-dir*>conf\tfsconfig.properties file.

b Uncomment #spEventPlugin= and provide the name for the filter's support plugin implementation:

```
spEventPlugin=com.hp.ov.selectfederation.filters.support.FilterSupport
Plugin
```

c Add the following line underneath, replacing .mycompany.com with the domain that makes sense according to your installation:

```
filterSupport.cookieDomain=.mycompany.com
```

> If the #spEventPlugin= line is not present because you have a previous version of the tfsconfig.properties file, then just add the two lines anywhere in the file.

d Optionally, you may set the following parameters, but the defaults are assumed if you choose not to provide the values:

```
# If not specified, this value is assumed to be "SFSession"
filterSupport.cookieName=SFSession

# If not specified, this value is assumed to be "-1"
filterSupport.cookieMaxAge=-1
```

e Restart the Application server that hosts the SP to enable the EventPlugin.

5 Deploy the <*cd-base-directory*>\filters\support\tfs-fs.war file to your Application Server that hosts your Select Federation war files.

- If your Select Federation install uses the built-in application server, deploy the tfs-fs.war file by placing it in the <*SF-installation-dir*>\webapps\ directory.
- If your Select Federation install uses WebLogic or WebSphere, deploy the tfs-fs.war file through the administrative console. See the respective application server documentation for details.

You are now ready to add the appropriate filter to your web server.

How Filter-Support Works

The following Filter-Support components are provided to facilitate the filter installation and configuration:

- **FilterSupportPlugin** — You can configure and use this plugin at your SP site installation as explained in the How to Configure Filter-Support on page 24. This plugin sets the cookie (used by filters) in the domain you specify after a user returns from authenticating with an IDP. (The domain should be shared by both the web server and the Select Federation application server.) Based on the cookie, the filter determines whether to allow users access to the resources or redirect them to a login URL.
- **FSService** — Uses a Select Federation session as a key to retrieve user-specific information from the SP databases for the filter.

- **tfs-fs.war** — Provides a login page (`login.jsp`), which can be configured as the login URL for either of the two filters (Apache or IIS) that you install. The login page allows the user to choose any one of the filter's SP's partner IDPs for being authenticated. The login page uses the following two URL parameters and the Select Federation APIs to do Single-Sign-On (SSO):
  - **targetURL** (required) -— The absolute URL of the page on the web server for which the user needs to authenticate. This URL is provided so that the user can be returned to the originating page on the web server as part of completing the SSO.
  - **targetIDP** (optional) — This URL is the providerId of the IDP to which the user should be directed.

By configuring Filter-Support, you enable the filter that is present at the web server to take the value of the cookie set by the FilterSupportPlugin to retrieve user information through the FSService, and to provide the headers to the filter's applications.

IIS Filter

The IIS filter and configuration interface for the IIS filter are implemented as Windows `DLL` files.

For the IIS filter, only Windows 2003 is supported.

These `DLL` files are installed and uninstalled by using the IIS console or by using scripts. Using the filter configuration interface, you can configure the filter with all required configurations (site level and virtual directory level) and store them in a metabase.

The following sections provide instructions for the IIS filter processes:

- Installing and Uninstalling the IIS Filter
- How to Configure the IIS Filter
- How to Use the IIS Filter
- IIS Filter Log File

Installing and Uninstalling the IIS Filter

The instructions in the following sections describe how to install and uninstall the IIS filter configuration interface and the IIS filter:

- How to Install the IIS Filter Configuration Interface
- How to Install the IIS Filter
- How to Uninstall the IIS Filter Configuration Interface
- How to Uninstall the IIS Filter

How to Install the IIS Filter Configuration Interface

When you install the IIS filter configuration interface, you are installing the DLL files required for virtual directory and site-level configurations.

Perform the following steps to install the IIS filter configuration interface:

1 Double-click on the **install.cmd** executable in `<cd-base-directory>\filters\iis\win32\conf\`.

The Open File - Security Warning dialog opens.

2 Click on the **Run** button.

You are prompted to open a filter file.

3 Click on the **Open** button for each file until all are opened.

4 Close the IIS console and restart it again.

The IIS Filter Configuration Interface is installed. Continue to install the IIS filter itself. See the next section, How to Install the IIS Filter, for instructions.

How to Install the IIS Filter

Before installing the IIS filter, you must have installed the IIS filter configuration interface (see How to Install the IIS Filter Configuration Interface on page 26).

You can install the IIS filter in one of two ways:

- Using the IIS console
- Executing a script

Installing the IIS Filter Using the IIS Console

Perform the following steps to install the IIS filter using the IIS console:

1 Save the `<cd-base-directory>\filters\iis\win32\SFFilter.dll` anywhere on your local hard drive.

2 Add the filter as follows:

a Select the Web Site you would like to protect in the left panel and right-click.

b Select the **Properties** menu option, then the **ISAPI Filters** tab in the Web Site Properties dialog.

c Click on the **Add** button.

The Add/Edit Filter Properties dialog opens.

d Enter the **Filter** name. For example, **SFFilter.**

e Enter the path name or browse for the **Executable** `SFFilter.dll` file you saved on your hard drive.

f Click **OK**.

3 Add the Web Service extension (for IIS 6.x only) as follows:

a Right-click on the **Web Service Extension** in the left panel.

b Select the **Add a new Web service extension...** option.

The New Web Service Extension dialog opens.

c Enter the Extension name. For example, **SFFilter.**

d Check the **Set extension status to Allowed** check box.

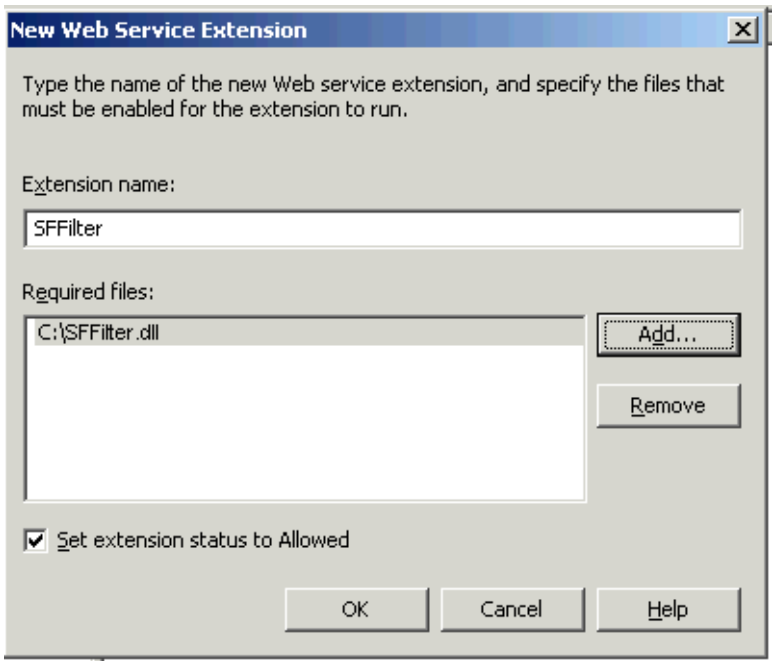
e Click on the **Add** button.

The Add file dialog opens.

f Enter the path name or browse for the **Executable** `SFFilter.dll` file you saved on your hard drive.

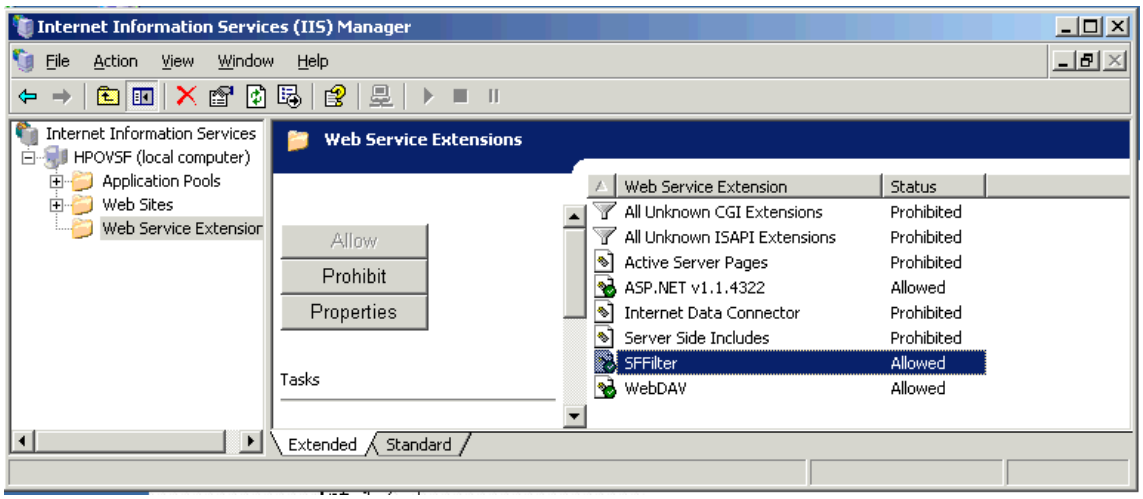
g Click **OK**.

The path displays in the Required files box in the New Web Service Extension dialog.



h Click **OK** again.

The Web Service Extension displays in the right panel with the status set to Allowed.



The IIS filter is installed.

4 After installing the IIS filter you must do the following steps or the filter may not load properly:

a Add <*cd-base-directory*>\filters\iis\win32\redist to the system PATH variable.

b Reboot the machine.

You are now ready to configure the IIS filter. See How to Configure the IIS Filter on page 30 for instructions.

Installing the IIS Filter Using a Script

Perform the following steps to install the IIS filter (both IIS 5.x and 6.x) using a script:

1 Save the <cd-base-directory>\filters\iis\win32\SFFilter.dll anywhere on your local hard drive.

2 Execute the Visual Basic script Install_SF_isapi_filter.vbs, such as:

Install_SF_isapi_filter.vbs <*SiteId*> <*FilterDLLFullPath*>.

For example:

Install_SF_isapi_filter.vbs 1 <location-on-hard-disk>\SFFilter.dll

The SiteId: 1, usually corresponds to the default web site, but the site you chose to protect may have a different SiteId. Be sure to determine which SiteId corresponds to your web site before running the script.

3 For IIS 6.0 users, add the web service extension by uncommenting the last two lines in the Install_SF_isapi_filter.vbs script.

The IIS filter is installed.

4 After installing the IIS filter you must do the following steps or the filter may not load properly:

a Add <*cd-base-directory*>\filters\iis\win32\redist to the system PATH variable.

b Reboot the machine.

You are now ready to configure the IIS filter. See How to Configure the IIS Filter on page 30 for instructions.

How to Uninstall the IIS Filter Configuration Interface

When you uninstall the IIS filter configuration interface, you uninstall the IIS filter configuration UI schema properties and the DLL files.

Perform the following steps to uninstall the IIS filter configuration interface:

1 Double-click on the uninstall.cmd executable in <*cd-base-directory*>\filters\iis\win32\conf\.

This command uninstalls the IIS filter configuration UI schema properties and the DLL files.

2 Close the IIS console and restart it again.

How to Uninstall the IIS Filter

You can uninstall the IIS filter in one of two ways:

- Using the IIS console

- Executing a script

Uninstalling the IIS Filter Using the IIS Console

Perform the following steps to uninstall the IIS filter using the IIS console:

1. Select the Web Site you chose to protect in the left panel in which you want to uninstall the IIS filter, and right-click.
2. Select the **Properties** menu option, then the **ISAPI Filters** tab in the Web Site Properties dialog.
3. Select the filter you want to uninstall such as the **SFFilter** you added.
4. Click on the **Remove** button and click **OK**.

   The filter is uninstalled.

Uninstalling the IIS Filter Using a Script

Perform the following steps to uninstall the IIS filter (both IIS 5.x and 6.x) using a script:

1. Execute the Visual Basic script `Remove_SF_isapi_filter.vbs`, such as:

   `Remove_SF_isapi_filter.vbs <SiteId> <FilterDLLFullPath>`.

   For example:

   `Remove_SF_isapi_filter.vbs 1 <`*`cd-base-directory`*`>\filters\iis\win32\SFFilter.dll`

   The `SiteId: 1`, usually corresponds to the default web site, but the site you chose to protect may have a different `SiteId`. Be sure to determine which `SiteId` corresponds to your web site before running the script.

   For IIS 6.0, you need to uncomment (if commented) the last two lines from the `Install_SF_isapi_filter.vbs` and `Remove_SF_isapi_filter.vbs` scripts.

   The filter is uninstalled.

How to Configure the IIS Filter

Before configuring the IIS filter, you must have installed the IIS filter configuration interface (see How to Install the IIS Filter Configuration Interface on page 26) and the IIS filter (see How to Install the IIS Filter on page 27).

There are two ways in which you can configure the filter using the filter configuration interface:

- At the web site level — This configuration is useful when you would like to protect access to all the virtual directories under the site using the IIS filter.
- At the virtual directory level — This configuration gives you the ability to do more fine-grained access control, so that each virtual directory can have its own filter configuration.

The IIS filter is configured by using the IIS filter configuration interface integrated with the IIS console.

Perform the following steps to configure the IIS filter:

1 Open the IIS Console as follows:

   a From the Start menu click **Run**.

   b In the Open text box, enter **inetmgr** and click **OK**.

2 Configure the IIS server or web site to be protected with server level configurations (cookie name, login page and so on):

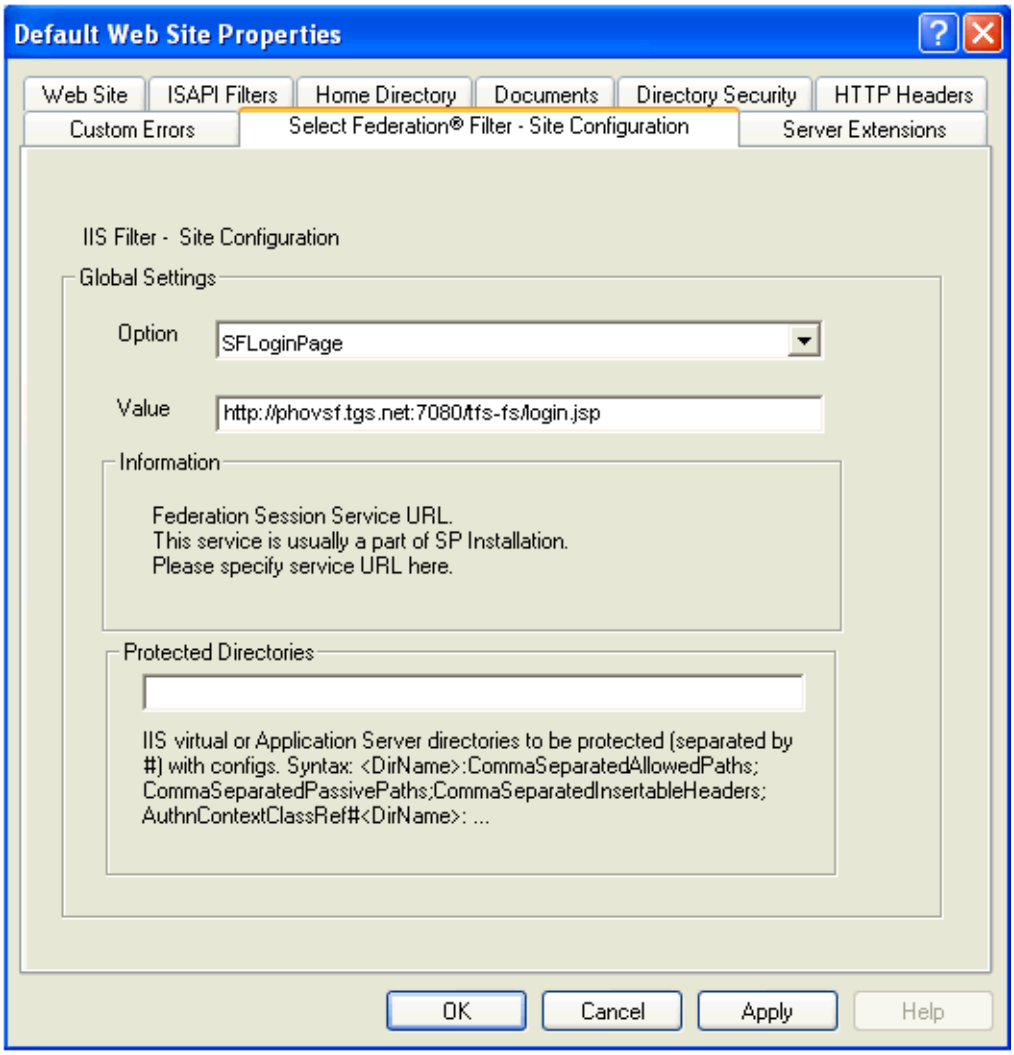
   a Select the web site you want to protect → Right click → Properties
   (On Windows 2000 and Windows 2003 servers, the same can be done as: Select IIS Server name (local computer) →Right click →Properties).

   The Web Site Properties dialog opens.

   b Select the **Select Federation Filter - Site Configuration** tab.

   c Configure the first property in the **Options** drop-down list by entering a value in the **Value** text box.

   Following is an example of configuring a property.



   d Click on **Apply** to save your setting.

e Continue configuring each of the properties listed in the **Option** drop-down list.

Click on **Apply** after entering a value for each configuration parameter so that your entries are saved correctly.

Option	Value
`SFLoginPage`	Provide a URL to a page where a user without the proper cookie for the Filter (to use as credentials) is redirected. You can use the `login.jsp` page provided in the `tfs-fs.war` file. For example: http://<sp-site>/tfs-fs/login.jsp or https://<sp-site>/tfs-fs/login.jsp
`SFFederationSessionService` (FSService)	Provide a URL to the FSService servlet. If you have finished the steps in How to Configure Filter-Support on page 24, the URL can be formed as either http://<sp-site>/tfs-fs/FSS or https://<sp-site>/tfs-fs/FSS depending on your SP deployment.
`SFLogFile`	Provide a location on your system where you feel it is appropriate for the Filter to log its messages. For example: `c:\Program Files\Select Federation\logs\SFFilter.log`. It is essential to specify the log file location since the log file is important for re-confirming the configuration used by the filter, understanding the filter functionality and gathering information to debug any issues. **Note**: If you are using a virtualization software to test the filter, be aware that the filter is not able to place the file at the specified location every time. You will need to figure out to which virtual directory the filter writes the file.
`SFDebug`	Provide the value as **TRUE** for the filter to log debug statements or **FALSE** not to log debug statements.
`SFCACerts` - (Certificate Authority Certificates)	Provide the file path name of the CA certificates for server validation.
`SFSkipServerAuth` - (Skip Server Authentication)	Provide the value as **True** to skip server authentication or **False** to authenticate. See Appendix C, Configuring Server and Client Authentication for more information.
`SFCookieName`	Provide the name of the cookie. The filter checks this in the request headers. The cookie name must be the same as the cookie set by the Event Plugin.

Option	Value
SFSkipClientAuth - (Skip Client Authentication)	Provide the value as **True** to skip client authentication or **False** to authenticate. See Appendix C, Configuring Server and Client Authentication for more information.
SFKeyFile	Provide the path name of the key file containing the private key to use along with the certificate.
SFCertFile (Certification File)	Provide the path name of the certification file that contains the client certificate to be used for client authentication.
SFPassPhrase	If the Key file is encrypted, provide the passphrase to use the key file.

f Click on **OK** to save the changes and close the window, or click on **Apply** to save the changes and keep the window open.

3 Configure virtual directories with directory level configurations:

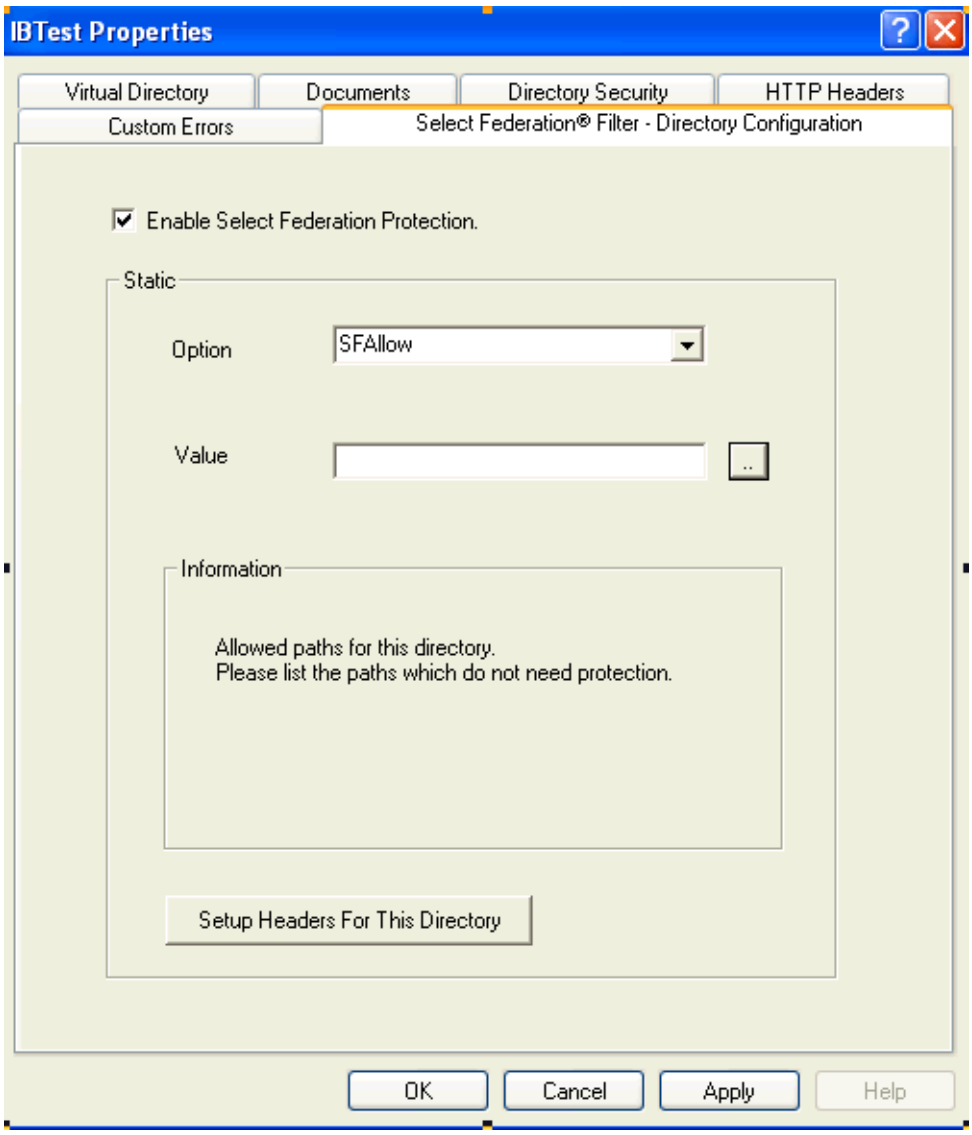
a Select a virtual directory in a web site.

b Right-click on the virtual directory and select the **Properties** menu option.

The web site Properties dialog opens.

c Select the **Select Federation Filter - Directory Configuration** tab.

d Check the **Enable Select Federation Protection** check box if you want to protect the virtual directory as shown in the following figure:


IBTest Properties
Virtual Directory
Documents
Directory Security
HTTP Headers
Custom Errors
Select Federation® Filter - Directory Configuration
Enable Select Federation Protection.
Static
Option
SFAllow
Value
..
Information
Allowed paths for this directory.
Please list the paths which do not need protection.
Setup Headers For This Directory
OK
Cancel
Apply
Help

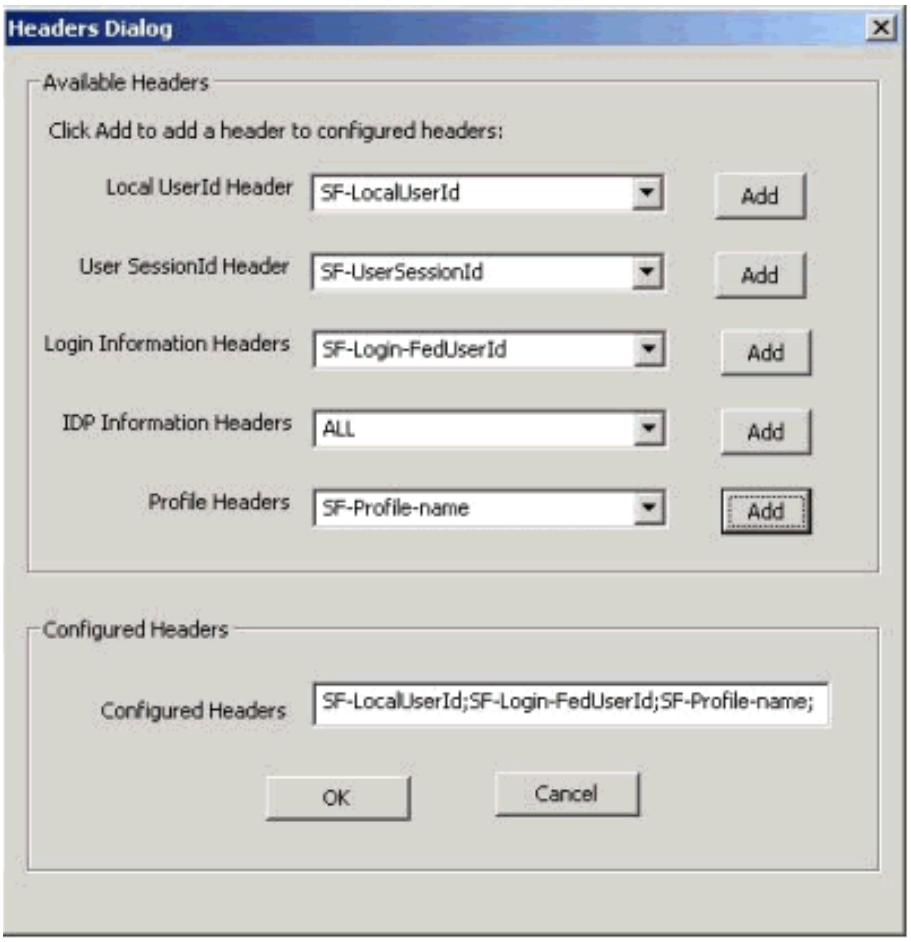
e Optionally, configure the following available properties in the **Option** drop-down list:

Option	Value
SFAllow	Provide a list of semicolon-separated relative URLs that should not be protected by the filter, in one of two ways: • Browse for the relative URLs by clicking the browse button (**..**) next to the Value input field. • Enter the relative URLs manually. For example, with URLs such as: http://superSecure.it.com/secureNow.asp and http://superSecure.it.com/secureLater.asp, the relative URLs would be entered as: **secureNow.asp;secureLater.asp;**
SFPassive	Provide a list of semicolon-separated relative URLs that should be treated as passive URLs (see Overview of Filters on page 23). The format is similar to the example given above for SFAllow.
SFAuthnContext	Choose an Authentication Context from the drop-down list, which serves as the minimum level of authentication needed to access the resources in the virtual directory. Note that the strength of the Authentication Context you configure should be less than or equal to the one specified in your SP's tfsconfig.properties configuration file. For example, if you select SmartCard in the filter configuration, and the default AuthnContext configured at the SP is PasswordProtectedTransport, you will not be granted access to the protected resource. However, if you specify Password in the filter configuration, you will be granted access. You can choose from the following values (listed in increasing levels of strength): • Password • PasswordProtectedTransport • MobileContract • MobileDigitalID • PreviousSession • Smartcard • SmartcardPKI • SoftwarePKI • TimeSyncToken If you do not specify a value for the authentication context, the default authentication context is used. This means that any level of authentication will be allowed.

f Optionally, click on the **Setup headers for this directory** button to customize which headers are set.

By default, all the information pertaining to the session, IDP, and user profile is made available as HTTP header variables, which you can customize.

The Headers Dialog opens.



g Select the header variables you wish to add and click the **Add** button after every selection.

The consolidated list displays in the **Configured Headers** section of the dialog box.

h Click on **OK** to save the changes and close the window, or click on **Apply** to save the changes and keep the window open.

4 Optionally, verify whether the properties are successfully added by doing one of the following:

a Select the properties of the virtual directory or server and open the Select Federation tab.

You should see the properties with the conf

igured values.

b You can use the MetaEdit tool to view the configured properties and their respective values. Download this tool from the following page and click the **Mtaedt22.exe** link:

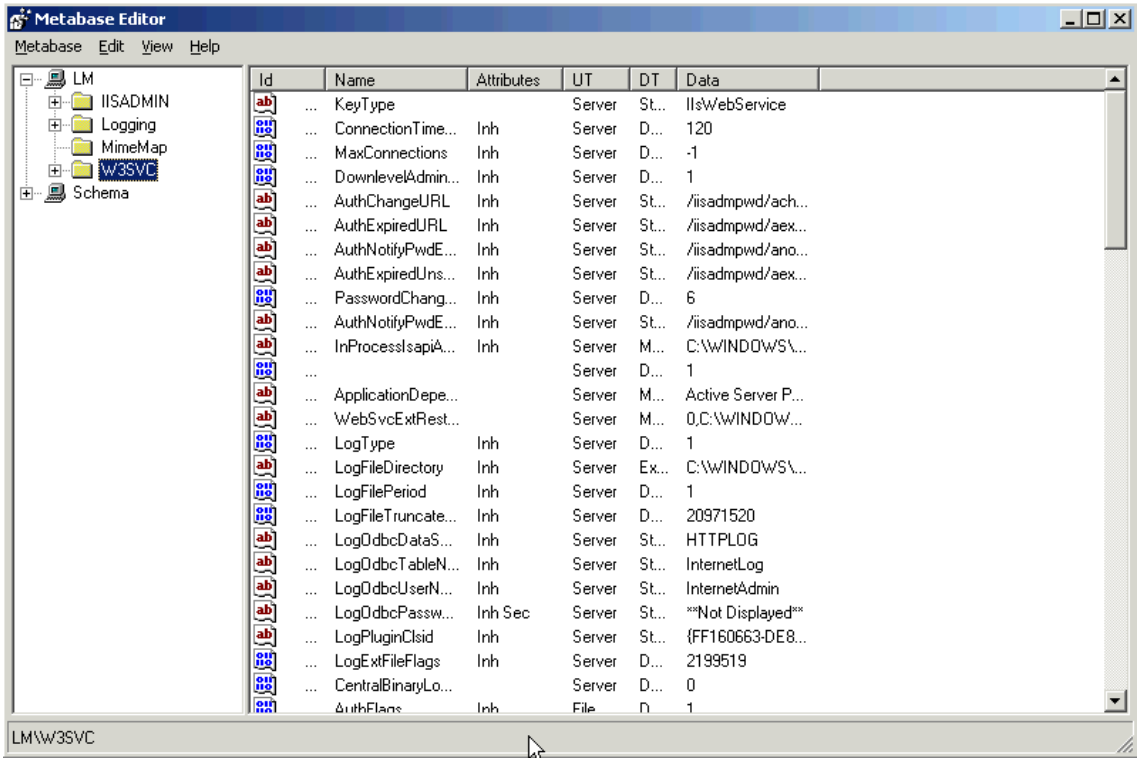
http://support.microsoft.com/default.aspx?scid=kb;en-us;301386&sd=tech

— To view the server level properties: select LM → W3SVC

— To view the web site level properties: select LM → W3SVC→ 1 → ROOT

— To view the virtual directory level properties: select LM → W3SVC→ 1→ROOT → <*DirName*>.

On the right-hand side you will see the configured properties under the Name column and their values under the Data column as shown in the following figure.



With IIS 6.0 (Win2K3) and IIS 5.0 (Win2K), you can do the server configurations either by right clicking on the server name (local computer) or specific web site you chose to protect and by selecting the properties menu. However, for IIS 5.1 (Win XP), the server configurations can ONLY be done by right clicking on the specific web site and selecting the properties menu. Because with IIS 5.1, we DO NOT get the properties menu after right clicking on the server name (local computer).

After making any changes to the filter settings using the configuration interface, do one of the following:

- It is recommended that you browse for the following dummy URL in order for the filter settings to take effect:

  http://<*sp-site*>/SFFilterConfigure.html

  The filter generates a `200 OK` response and sends a page with the following text to the client:

  ```
  SF filter configured!
  ```

- You can verify that the configuration changes have taken effect by looking at the filter log file.

How to Use the IIS Filter

This section provides an IIS filter sample to help you know how to use the IIS filter. This filter sample is provided as an example of how the filter may be used to retrieve information about the federated session, user profile, and partner IDP. This sample is very basic in that it prints all the headers that have been configured for that particular site or virtual directory.

Be sure that you have installed a partner IDP, which is configured to use an Access Management system or a directory server. The IDP is required to authenticate a user that initiates a federation from the SP.

Filter Sample

To access the sample, perform the following steps:

1. Copy the `getheaders.asp` sample from the *<cd-base-directory>*`\filters\samples\` directory to your protected site or virtual directory.
2. Open the `getheaders.asp` file and search for `SFRoot` to set `it` to the root URL of your Select Federation Application server SP instance.
3. Save your changes.
4. Now try to access the sample. For example:

   http://<iis_servername>/<protectedvirtual_directory>/getheaders.asp

   The browser should now be redirected to a login page, as configured in the Select Federation filter configuration interface (see Step 2 in How to Configure the IIS Filter on page 30).
5. Select the IDP that you would like to authenticate with from the drop-down list and click the **Login** button.

   The IDP login page opens where you are asked to enter your credentials.
6. Login with any user account that exists in the access management system or directory server that has been configured at the IDP.

   Once you as the user is authenticated by the IDP, the IIS filter then calls the FSService servlet to get the user information. The FSService should have been configured in the Select Federation filter configuration interface as a URL value for `SFFederationSessionService` (see Step 2 in How to Configure the IIS Filter on page 30).

   The request is then forwarded to the desired page: `getheaders.asp` in this sample, which prints out all the headers it receives.

IIS Filter Log File

The IIS filter log file path can be configured using the configuration interface. If the log file is not configured from the interface, then the filter looks for the `SFHome` registry entry under `HKEY_LOCAL_MACHINE` → `SOFTWARE` → `\Hewlett-Packard\OpenView` and creates the `Server\Logs\SFFilter.log` file relative to `SFHome`.

If the SFHome registry entry is not found, the IIS filter logs details into the C:\SFFilter.log file. This generated log file can be used to see the request processing details, headers set and cookie information.

If you set SFDebug to True in the Select Federation configuration interface, then the filter will output debug statements to the log file. This will help gain a better understanding of the inner working of the filter and help with troubleshooting.

Following is an example of a log file with the filter configuration information and request processing details.

```
Filter Configuration Details:
-----------------------------
Server configs:
ookieName=SFSession
LoginPage=http://localhost:10432/fs/login.jsp
FederationSessionService=http://localhost:10432/fs/fss.jsp
Debug=TRUE
SkipClientAuth=TRUE
CertFile=
KeyFile=
PassPhrase=
SkipServerAuth=TRUE
CACerts=
Virtual directory configs:
   /SFTest/:index.html;;SF-IDP-Home,SF-Profile-name-firstname,;

Protected URL: /SFTest/getheaders.asp
Cookie-Header:ASPSESSIONIDAARSDBSD=HNOJGPMCAPGJIBFAOKCONDDF;
SFSession=2dc4d67ae068b6979bf9f8097239e8b14eab1f2c
Expected cookie found
Successfully received info from Federation Session Service...
Cached info for sessionId=2dc4d67ae068b6979bf9f8097239e8b14eab1f2c
User information is set in the headers
SF-IDP-Home:null SF-Profile-name-firstname:testname
```

Apache Filter

The Apache filter is implemented as a shared library (.DLL for Windows or .SO for Linux).

For the Apache filter, the following platforms are supported:

- Windows 2000 and 2003
- Red Hat Linux AS 3.0 and 4.0

You install the Apache filter by copying the filter shared library into the <*apache_home*>/modules> directory. You then configure the Apache filter by modifying the httpd.conf file. If you need to change default settings, you can modify the filter.conf file.

Before you install and configure the Apache filter, be sure to carefully look its prerequisites.

The Select Federation Apache 2.0 module binaries are compatible with specific versions of Apache, but not with Apache 1.3. For details on supported Apache versions, see Appendix B, Supported Apache Versions.

The following sections provide instructions for the Apache filter processes:

- How to Install the Apache Filter
- How to Use the Apache Filter
- Apache Filter Log File

How to Install the Apache Filter

Complete the following steps to install the Apache filter on Windows and Linux platforms.

On Windows Platforms

Perform the following steps to install the Apache filter on Windows platforms:

1. Copy the Apache filter binary from `<cd-base-directory>/filters/apache/win32/SFModule.dll` to the `<apache_home>/modules/` directory.
2. Copy the files present in the `<cd-base-directory>/filters/apache/win32/redist/` folder into a local folder such that they will be in the system PATH. Or copy them into a new local folder and edit the system PATH to point to it as well.
3. Add the following line to the `<apache_home>/conf/httpd.conf` file:

   ```
   LoadModule SFModule_module modules/SFModule.dll
   ```

4. Copy the sample configuration file `<cd-base-directory>/filters/apache/conf/filter.conf` to the `<apache_home>/conf/` directory.
5. Restart your Apache web server.

On Linux Platforms

Perform the following steps to install the Apache filter on Linux platforms:

1. Configure Filter-Support as described in How to Configure Filter-Support on page 24.
2. Copy the Apache filter binary from `<cd-base-directory>/filters/apache/linux/SFModule.so` to the `<apache_home>/modules/` directory.

   The Apache filter binary includes the following run-time dependencies:

   - Apache APR
   - `OpenSSL`
   - `libcurl` (built with `OpenSSL`)
   - `libxml2`

3. Check the run-time dependencies of the Apache filter module to be sure all required components are present.

   a. Enter the following command at the prompt (#):

   `# ldd <apache_home>/modules/SFModule.so`

   The output may look something like the following:

   ```
   libxml2.so.2 => /usr/lib/libxml2.so.2 (0xb74bb000)
   libz.so.1 => /usr/lib/libz.so.1 (0xb74ad000)
   libcurl.so.3 => not found
   libdl.so.2 => /lib/libdl.so.2 (0xb74aa000)
   libstdc++-libc6.2-2.so.3 => /usr/lib/libstdc++-libc6.2-2.so.3 (0xb7468000)
   libm.so.6 => /lib/tls/libm.so.6 (0xb7446000)
   ```

```
libc.so.6 => /lib/tls/libc.so.6 (0xb730e000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0xb72fe000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0xb72f5000)
```

Any `not found` results indicate that either the dependencies are not installed, or do not exist in the linker path.

b Enter the following command at the prompt (#) to copy the missing components provided in the *<cd-base-directory>*`/filters/apache/linux/redist/` directory to a new local directory:

```
# cp -r <cd-base-directory>/filters/apache/linux/redist/ <local-dir>/filters/
```

c Add the path of the new local directory (*<local-dir>*`/filters/redist`) to the `/etc/ld.so.conf` file.

d Execute the following command at the prompt (#):

`# ldconfig`

You might see some warnings such as:

```
ldconfig: /var/tmp/filters/redist/libxml2.so.2 is not a symbolic link
ldconfig: /var/tmp/filters/redist/libcurl.so.3 is not a symbolic link
ldconfig: /var/tmp/filters/redist/libapr-1.so.0 is not a symbolic link
```

Ignore these warnings and proceed to the next step.

e Enter the following command at the prompt again:

`#ldd <apache_home>/modules/SFModule.so`

This time, the results should show that all dependencies are satisfied.

```
libxml2.so.2 => /var/tmp/filters/redist/libxml2.so.2 (0xb74a4000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7496000)
libcurl.so.3 => /var/tmp/filters/redist/libcurl.so.3 (0xb7464000)
libdl.so.2 => /lib/libdl.so.2 (0xb7461000)
libstdc++-libc6.2-2.so.3 => /usr/lib/libstdc++-libc6.2-2.so.3 (0xb741f000)
libm.so.6 => /lib/tls/libm.so.6 (0xb73fd000)
libc.so.6 => /lib/tls/libc.so.6 (0xb72c5000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0xb72b5000)
libssl.so.0.9.8 => /var/tmp/filters/redist/libssl.so.0.9.8 (0xb727a000)
libcrypto.so.0.9.8 => /var/tmp/filters/redist/libcrypto.so.0.9.8 (0xb7155000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0xb714c000)
```

If `libstdc++-libc6.2-2.so.3` is not found, you need to install the `rpm` appropriate to your system to satisfy this dependency.

4 Add the following line to the *<apache_home>*`/conf/httpd.conf` file:

`LoadModule SFModule_module modules/SFModule.so`

5 Restart your Apache web server.

6 Copy the sample configuration file *<cd-base-directory>*`/filters/apache/conf/filter.conf` to the *<apache_home>*`/conf/` directory.

How to Configure the Apache Filter

Before configuring the Apache filter, you must have installed the Apache filter (see How to Install the Apache Filter on page 40).

Configure the Apache filter by completing the following tasks:

- Task 1: Edit the <apache_home>/conf/filter.conf file to configure the filter-specific options to match your SP installation:

  The following filter-specific options are shared across all directories:

  - `CookieName`
  - `LoginPage`
  - `FederationSessionService`
  - `Debug`
  - `SkipClientAuth`
  - `CertFile`
  - `KeyFile`
  - `PassPhrase`
  - `SkipServerAuth`
  - `CACerts`

- Task 2: Edit the <apache_home>/conf/httpd.conf file to configure the directories and locations that you wish to protect:

Task 1: Edit the <apache_home>/conf/filter.conf file to configure the filter-specific options to match your SP installation:

Provide values for the following filter-specific options to match your SP installation:

Uncomment the following line:	**Provide the following value:**
`#CookieName=..`	The same cookie name as the one you configured for the `filterSupport.cookieName` parameter in the `tfsconfig.properties` file. (See How to Configure Filter-Support on page 24.)
`#LoginPage=...`	A URL to a page where a user without the proper cookie for the filter (to use as credentials) is redirected. You can use the `login.jsp` page provided in the `tfs-fs.war` file. For example: http://<sp-site>/tfs-fs/login.jsp or https://<sp-site>/tfs-fs/login.jsp
`#FederationSessionService=...`	A URL to the FSService servlet which is called by the filter to get LoginInfo, IDPInfo, User ProfileInfo and so on. If you have finished the steps in How to Configure Filter-Support, the URL can be formed as either http://<sp-site>/tfs-fs/FSS or https://<sp-site>/tfs-fs/FSS depending on your SP deployment.
`#Debug=...`	**TRUE** for the filter to log debug statements or **FALSE** not to log debug statements.
`#SkipServerAuth=...`	**TRUE** to skip server authentication or **FALSE** to authenticate. See Appendix C, Configuring Server and Client Authentication for more information.
`#CACerts=...`	The path to the file that contains the CA certificates for server validation.
`#SkipClientAuth=...`	**TRUE** to skip client authentication or **FALSE** to perform client authentication for https communication. See Appendix C, Configuring Server and Client Authentication for more information.
`#CertFile=...`	The path to the file that contains the client certificate, to be used for client authentication during the SSL Handshake. The certificate should be in PEM format.
`#KeyFile=...`	The path to the file that contains the private key, to be used with the client certificate for client authentication during the SSL Handshake.
`#PassPhrase=...`	If the key file is encrypted, provide the passphrase to use the key file.

Task 2: Edit the <apache_home>/conf/httpd.conf file to configure the directories and locations that you wish to protect:

1. Add the `SFProtect` option to the specified directories to register them with the filter.

```
<Directory "/path/to/directory">
   ... Other Standard Options ...
   SFProtect
   ...
</Directory>
```

2. Specify the headers for the values to be retrieved:

```
<Directory "/path/to/directory">
   ... Other Standard Options ...
   SFProtect
   SFHeaders SF-LocalUserId,SF-UserSessionId,SF-Login,SF-IDP,SF-Profile
   ...
</Directory>
```

The above configuration for `SFHeaders` is equivalent to specifying all the attributes individually, for example:

```
SFHeaders SF-Login-IdpFedUserId,SF-Login-AuthnContextClassRef,SF-
Login-AuthnInstant,SF-Login-ReauthOnOrAfter,SF-IDP-IdpProviderId,SF
-IDP-Home,SF-IDP-Name,SF-IDP-Description,SF-IDP-LogoRef,SF-IDP-Logo
Text,SF-Profile-name,SF-Profile-name-title,SF-Profile-name-firstnam
e,SF-Profile-name-lastname,SF-Profile-home-street,SF-Profile-home-c
ity,SF-Profile-home-state,SF-Profile-home-country,SF-Profile-home-p
ostalCode,SF-Profile-personal-email,SF-Profile-personal-phone,SF-Pr
ofile-work-street,SF-Profile-work-city,SF-Profile-workstate,SF-Prof
ile-work-country,SF-Profile-work-postalCode,SF-Profile-work-email,
SF-Profile-work-phone
```

See Appendix A, How Filters Work for more details.

3. Specify the resources that should not be protected by the filter:

```
<Directory "/path/to/directory">
   ... Other Standard Options ...
   SFProtect
   SFHeaders...
   SFAllow allow.html,allow2.html
   ...
</Directory>
```

4. Specify the resources that should be passive:

```
<Directory "/path/to/directory">
   ... Other Standard Options ...
   SFProtect
   SFHeaders...
   SFAllow...
   SFPassive passiveurl1,passiveurl2
   ...
</Directory>
```

5 Optionally, specify a value for the Authentication Context to be the minimum level of authentication needed to access the resource.

The strength of the Authentication Context you configure should be less than or equal to the one specified in your SP's `tfsconfig.properties` configuration file.

For example, if you select `SmartCard` in the filter configuration, and the default AuthnContext configured at the SP is `PasswordProtectedTransport`, you will not be granted access to the protected resource. However, if you specify `Password` in the filter configuration, you will be granted access.

You can choose from the following values (listed in increasing levels of strength):

- `Password`
- `PasswordProtectedTransport`
- `MobileContract`
- `MobileDigitalID`
- `PreviousSession`
- `Smartcard`
- `SmartcardPKI`
- `SoftwarePKI`
- `TimeSyncToken`

If you do not specify a value for the authentication context, the default authentication context is used. This means that any level of authentication will be allowed.

6 Specify an alternate Authentication context if you are not satisfied by the default one (usually you would not have a need to specify this):

```
<Directory "/path/to/directory">
   ...Other Standard Options ...
   SFProtect
   SFHeaders...
   SFAllow...
   SFPassive...
   SFAuthContext minContext
   ...
</Directory>
```

7 Specify a line such as `SFPassive passive.*, which` marks all the paths that begin with passive as passive paths.

How to Use the Apache Filter

This section is describing an Apache filter sample to help you know how to use the Apache filter. This filter sample is provided as an example of how the filter may be used to retrieve information about the federated session, user profile, and partner IDP. This sample is very basic in that it displays all the headers that have been configured for that particular site or virtual directory.

Be sure that you have installed a partner IDP, which is configured to use an Access Management system or a directory server. The IDP is required to authenticate a user that initiates a federation from the SP.

Filter Sample

To access and use the sample, perform the following steps:

1. Copy the `getheaders.php` sample from the `<cd-base-directory>/filters/samples/` directory to your protected site or protected directory.
2. Open the `getheaders.php` file and search for `SFRoot` to set it to the root URL of your Select Federation Application server SP instance.
3. Save your changes.
4. Now try to access the sample. For example:

   http:// <*servername:port*>/<protectedDirectory>/getheaders.php

   The browser should now be redirected to a login page, as configured in the Select Federation filter configuration file (see How to Configure the Apache Filter on page 42).
5. Select the IDP that you would like to authenticate with from the drop-down list and click the **Login** button.

   The IDP login page opens where you are asked to enter your credentials.
6. Login with any user account that exists in the access management system or directory server that has been configured at the IDP.

   Once you as the user is authenticated by the IDP, the Apache filter then calls the FSService servlet to get the user information. The FSService should have been configured in the Select Federation filter configuration file as a URL value for `SFFederationSessionService` (see How to Configure the Apache Filter on page 42).

   The request is then forwarded to the desired page: `getheaders.php` in this sample, which displays all the headers it receives.
7. If `php` is not enabled, check the `<apache-root>/logs/error.log` log file. The filter logs debug (`if DEBUG=TRUE`) and normal log messages into a separate file `filter.log` in the `logs/` directory.

   This makes it easier to see the filter- specific logs. Fatal errors go into the Apache `error_log` file.

Apache Filter Log File

The Apache filter module logs its messages into the `<apache_home>/logs/filter.log` file. If there are any problems in generating the log file, the messages are logged to the Apache error log file. For example: `<apache_home>/logs/error_log file`.

Following is an example of a log file with the filter configuration information and request processing details:

```
[SFFilter.init] Total directories Configured = 2.
[SFFilter.init] Cookie Name : SFSession.
[SFFilter.init] Login Page : https://sp1.HPOVSF.tgx.net:7443/tfs-fs/
login.jsp.
[SFFilter.init] Federation Session Service : https://sp1.HPOVSF.tgx.net:7443/
tfs-fs/FSS.
[SFFilter.init] Client Certificate File Path : newcert.pem.
[SFFilter.init] Client Key File Path : newreq.pem.
[SFFilter.init] Key File Passprhase : NO_PWD.
[SFFilter.init] CA Certificates File : /usr/local/test-scripts/trustedSP.pem.
[SFFilter.init] Skip Client Authentication : TRUE.
```

```
[SFFilter.init] Skip Server Authentication : FALSE.
[SFFilter.init] Debug : TRUE
[DEBUG: 07/Apr/2006:12:46:23 AM PDT] Match found at /sffilter/.*
[LOG: 07/Apr/2006:12:46:23 AM PDT]  Found Protected Url /sffilter/
getheaders1.php
[DEBUG: 07/Apr/2006:12:46:23 AM PDT] Did not match any Allowed Path : (/
sffilter/getheaders1.php)
[LOG: 07/Apr/2006:12:46:23 AM PDT]  Session ID Not Found.
[DEBUG: 07/Apr/2006:12:46:23 AM PDT] Did not match any Passive Path : (/
sffilter/getheaders1.php)
[LOG: 07/Apr/2006:12:46:23 AM PDT]  Redirected to sign-in page.
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Match found at /sffilter/.*
[LOG: 07/Apr/2006:12:46:37 AM PDT]  Found Protected Url /sffilter/
getheaders1.php
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Did not match any Allowed Path : (/
sffilter/getheaders1.php)
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Cookie Value
:SFSession=76ce83b37d20934e0d7d90e2760338e31510f62c
[LOG: 07/Apr/2006:12:46:37 AM PDT]  Found SessionId
(SFSession=76ce83b37d20934e0d7d90e2760338e31510f62c)
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Fetching Resource: (https://
sp1.HPOVSF.tgx.net:7443/tfs-fs/
FSS?sessionId=76ce83b37d20934e0d7d90e2760338e31510f62c)
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Status:Success
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Following headers are available.
-------
[DEBUG: 07/Apr/2006:12:46:37 AM PDT]
LocalUserId:d3d6a8fd470c3b1e7f0083cb5179b202f507a31c
[DEBUG: 07/Apr/2006:12:46:37 AM PDT]
UserSessionId:76ce83b37d20934e0d7d90e2760338e31510f62c
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Name:idp1
[DEBUG: 07/Apr/2006:12:46:37 AM PDT] Description:
....
```

5 Select Federation API Overview

This chapter describes how to access the HP OpenView 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 7, API Samples.

API Documentation

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

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

Alternatively, the documentation is also available through the `Programmers Reference` link under the `Programmer Tasks` section of the Select Federation management console. The Select Federation management console is typically deployed at:

http://<*base-url*>/tfs-internal/

Where the **<*base-url*>** is the base URL of the application server on which you have deployed Select Federation.

Each sample on the Select Federation CD, located in the `<cd-base-directory>/api/samples/` directory, demonstrates the use of one aspect of the Select Federation APIs. See Chapter 7, 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

- IDP Directory Plugin Interface

See API Documentation on page 49 for information on accessing the API documentation and the Chapter 7, 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 67.

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 49 for information on accessing the API documentation and the Chapter 7, 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

HP OpenView 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.

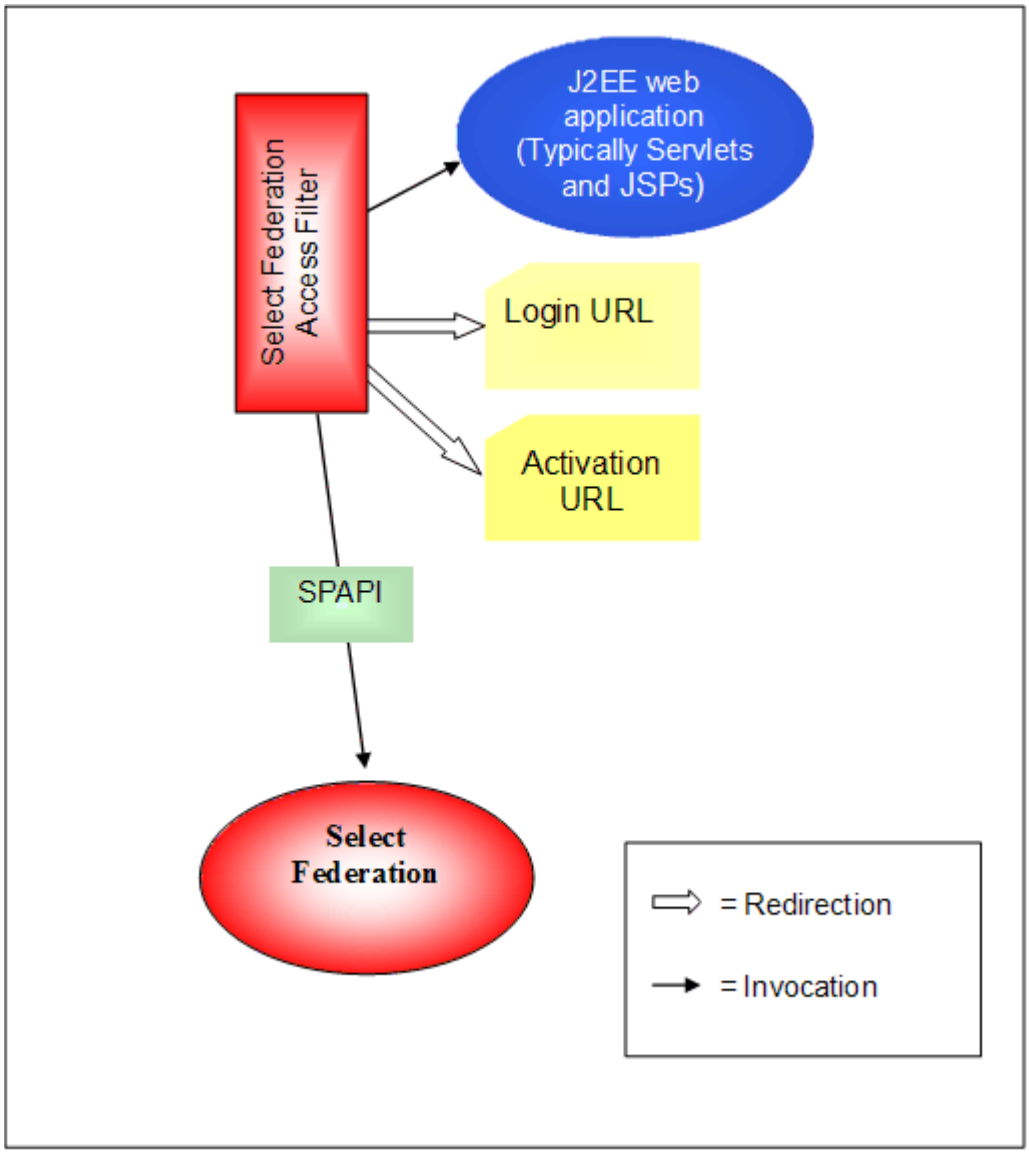
The Access filter maintains a session attribute called spUser in the J2EE session of the user. If this attribute is not set, the user is not expected to be logged in.

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



See API Documentation on page 49 for information on accessing the API documentation and the Chapter 7, API Samples for descriptions of the samples on the Select Federation CD.

6 Activation and Auto Enrollment

HP OpenView 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*.

7 API Samples

This chapter provides descriptions of the web application samples provided on the HP OpenView 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 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 proxying 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:

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

4 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.

5 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.

6 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 idpAuthnPlugin 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 via 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 don't see the profile parameters, the IDP probably doesn't 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 via the IDP) with an existing local account (enter any user ID here) or the sample can generate a unique user ID on-the-fly (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 Homepage 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 Homepage 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 simply navigate using a browser to the `/idp-portal` URL of that application server.

IDP-Intro

This sample describes the use of the Liberty introduction protocol to find out the IDP that a user belongs to. 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 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 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 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 on the federate link for the configured IDP that you wish to federate the SP account with. You will be 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 on the Read Introduction Cookie link.
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 the logout link if needed.
2. Click on the link to login via 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 on terminate federation for the IDP federation that you wish to terminate.

IDP-Proxy

This sample demonstrates the use of the proxying 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. This sample requires that Select Federation is configured to do http based logout and not SOAP based logout.

Sources

- **`web/login.jsp`:** Provides a way for the user to login using the home IDP or log in 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 logout the user from the home IDP.

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, as follows:

   `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. Download the meta-data for all three sites as the Liberty 1.2 combined IDP+SP meta-data.
4. Upload the IDP2 meta-data into IDP1 as an Application (SP) site.
5. Upload the IDP1 meta-data into IDP2 as an Authority (IDP) site.
6. Upload the IDP2 meta-data into SP1 as an Authority (IDP) site.
7. Upload the SP1 meta-data into IDP2 as an Application (SP) site.
8. Deploy the sample idp-sample on IDP1.
9. Deploy the sample idp-proxy on IDP2.
10. Deploy the sample sp-sample on SP1.
11. Navigate to the URL for the `sp-sample` and click on the link **Login via <proxy IDP>**.
12. On the next login page that you see, click on the link **Login via <home IDP>**.
13. On the home IDP login page, enter a username (no password required) and you will see the activation or logged in page of the `sp-sample`.
14. Click on the logout link and you will be logged out of both IDPs.

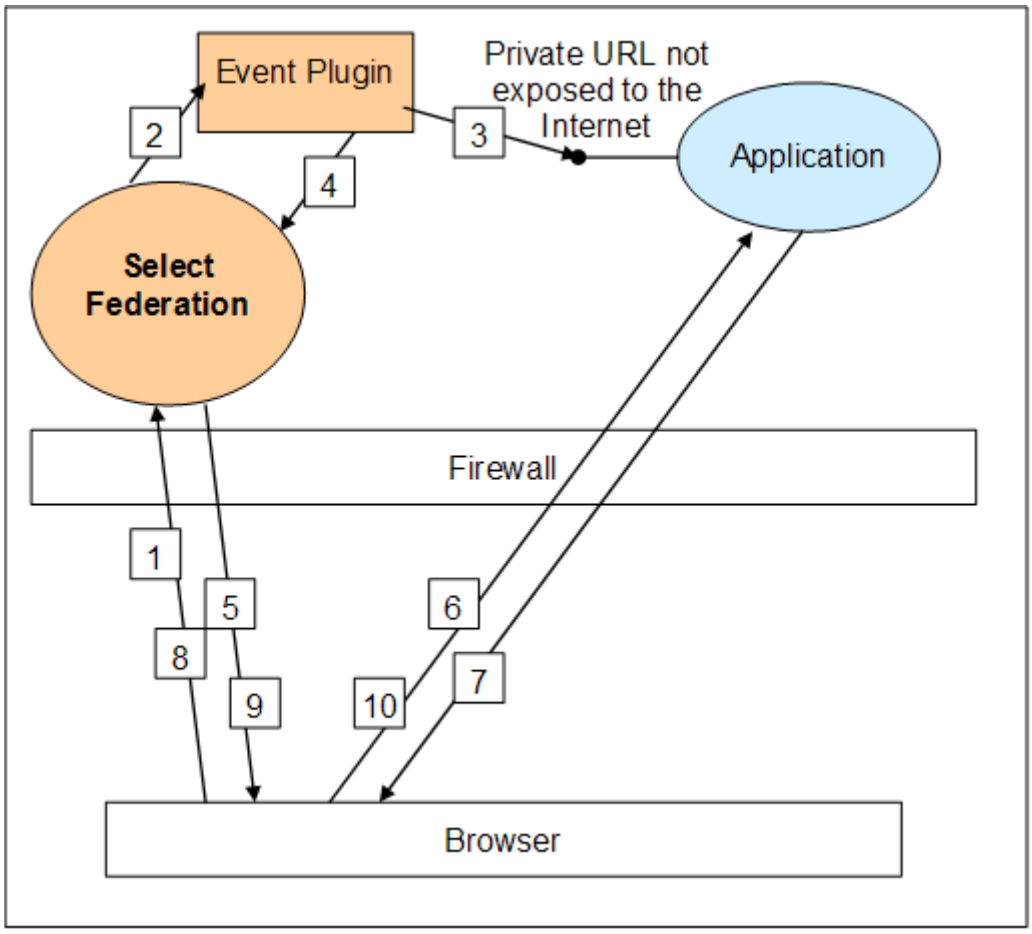
sp-SampleEventPlugin

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 diagram below:

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 via 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-SPEventPlugin Sample

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

To run the sp-EventPlugin 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 IDP 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 2 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.
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.

A How Filters Work

The filters are configured to protect (allow access only after authentication) certain directories, allow access to certain files without authentication. It is also possible to configure what information (login event, IDP details from which a user is authenticated, user profile attributes and so on) should be made available to the applications. Other filter configuration parameters are cookie name, login page location, federation session service URL, certificate and key details for client authentication and configurations for each protected directory.

The IIS filter reads the configurations from the metabase using the ADSI (Active Directory Service Interface). This filter reads the configurations either when it receives its first request after the IIS restart or by browsing the configuration URL `<BaseURL>/SFFilterConfigure.html`. The Apache filter reads all the configuration details from the configuration file when the Apache web server starts.

When a filter receives the HTTP request, it first compares the request URL with the protected URLs (can be accessed only by authenticated users). If the request URL is not protected then it does not do any additional processing and passes the unmodified request along to the web server (or the next filter in the chain).

If the request URL is protected, then it retrieves the cookie header from the request headers and looks for a particular cookie in the cookie header. After the user is authenticated, this cookie is set for the domain shared between the web server and the Select Federation installation with an ID to the user federation session Event-Plugin running on an SP site (see SP Event Plugin Interface on page 50 for more information). In the remainder of this chapter, the cookie is referred to as the “SF cookie”.

If the SF cookie is not present, the user is not authenticated and the filter redirects the user to the login page with the following as URL parameters:

- Original request URL
- Whether request URL is passive (URLs 1 or 0 depending upon the request URL is configured as passive)
- Authentication Context Class Reference (such as Password, Token, SIM, Certificate).

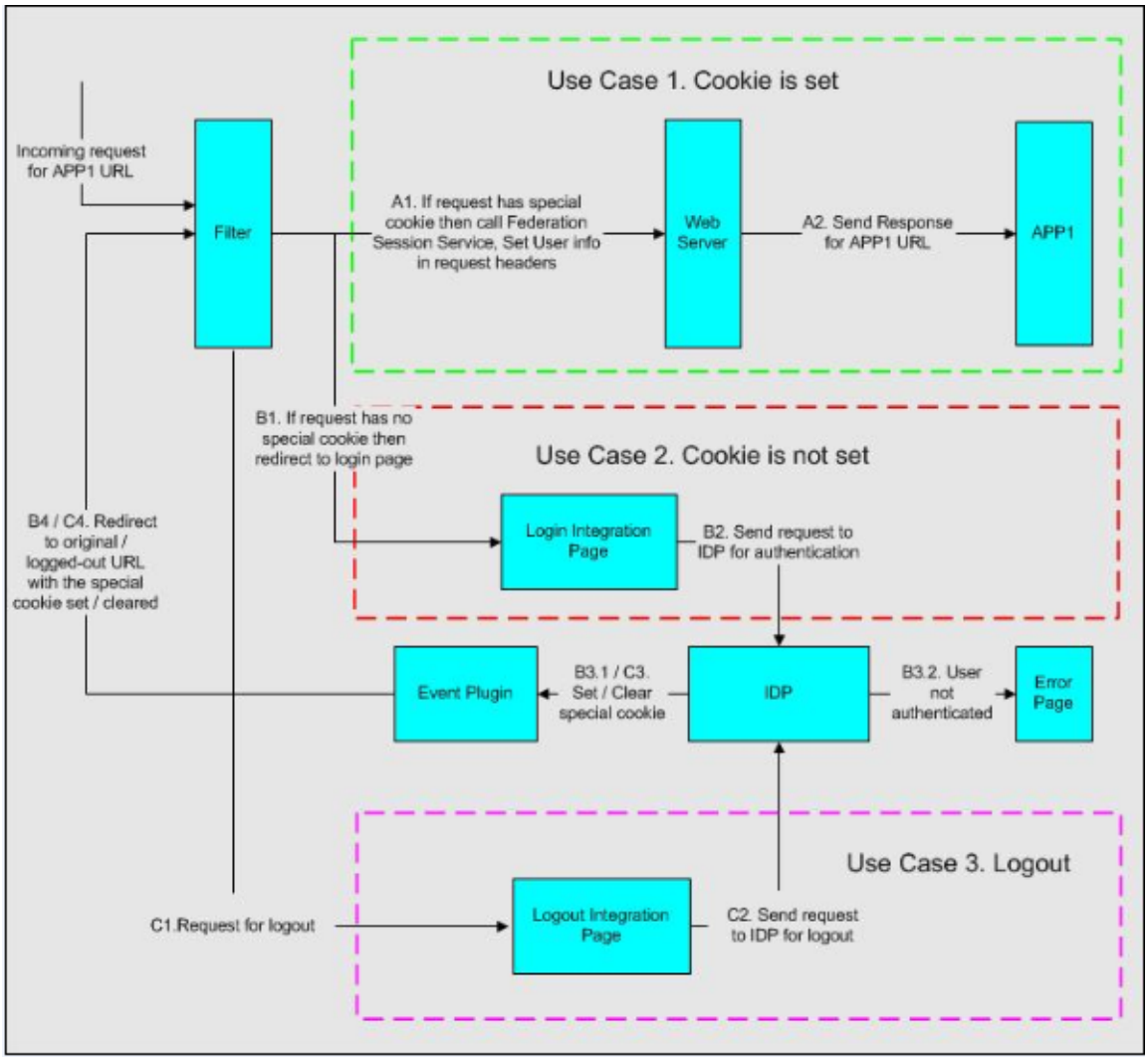
The filter maintains an in-memory cache of user or principal information indexed by sessionId to ensure high performance. If the filter finds a SF cookie in the cookie header, it gets the value (which happens to be the sessionId) and tries to find it in the cache. If the entry is found then it verifies that entry is valid (cached not before 15 minutes). Filter also refreshes (erases entries which are cached before 15 minutes) the entire cache after every 150 requests served. At any instance, the cache can have maximum 3000 valid entries and filter does not add the new entry if the cache is full. If entry is valid then it sets the request headers with principal information and lets the web server to process the request further.

In case the entry for `sessionId` is not found in the cache or entry is expired, the filter calls the federation session service (`<BaseURL>/tfs-fs/FSS` installed on SP site) with `sessionId` as the URL parameter. The federation session service returns the XML document with information about the principal.

The Filter then parses the XML document using libxml – XML parser library. If the "status" field value from the XML document is not "SUCCESS" (case like federation session service not available or invalid session-id parameter passed to federation session service etc) then it logs error message into the log file and redirects the user back to login page if the original request URL is protected. In this failure case, we redirect to login page because we treat this case as if user is not authenticated and invalid session-id parameter was passed.

If the filter receives the principal information successfully then it adds all the principal information in the cache indexed by `sessionId`. After this filter sets the request headers (as per the configurations made by the user) with the principal information and lets the web server process the rest of the request.

Figure 4 Filter Architecture



To make use of headers set by the filter, you need to know the following header names:

- `SF-LocalUserId`
- `SF-UserSession-Id`
- `SF-Login`: When you configure the Apache or the IIS Filters, you can either specify `SF-Login` to retreive all the `LoginInfo`-related headers or specify each one of the following individually:

- SF-Login-IdpFedUserId
- SF-Login-AuthnContextClassRef
- SF-Login-AuthnInstant
- SF-Login-ReauthOnOrAfter

- SF-IDP: When you configure the Apache or the IIS Filters, you can either specify SF-IDP to retreive all the IDPInfo-related headers or specify each one of the following individually:

  - SF-IDP-IdpProviderId
  - SF-IDP-Home
  - SF-IDP-Name
  - SF-IDP-Description
  - SF-IDP-LogoRef
  - SF-IDP-LogoText

- SF-Profile: When you configure the Apache or the IIS Filters, you can either specify SF-Profile to retreive all the ProfileInfo-related headers or specify each one of the following individually (SF-Profile-<*AttributeName*> where <*AttributeName*> is the actual attribute name):

  - SF-Profile-name
  - SF-Profile-name-title
  - SF-Profile-name-firstname
  - SF-Profile-name-lastname
  - SF-Profile-home-street
  - SF-Profile-home-city
  - SF-Profile-home-state
  - SF-Profile-home-country
  - SF-Profile-home-postalCode
  - SF-Profile-personal-email
  - SF-Profile-personal-phone
  - SF-Profile-work-street
  - SF-Profile-work-city
  - SF-Profile-work-state
  - SF-Profile-work-country
  - SF-Profile-work-postalCode
  - SF-Profile-work-email
  - SF-Profile-work-phone

  This SF-Profile list assumes that the tfsconfig.properties file for the SP, includes the following attributes:

  ```
  userAttrs=name name_title name_firstname name_lastname
  home_street home_city home_state home_country home_postalCode
  personal_email personal_phone work_street work_city work_state
  work_country work_postalCode work_email work_phone
  ```

Notice that the login information headers are prefixed with SF-Login-, IDP information headers are prefixed with SF-IDP- and profile information headers are prefixed with SF-Profile-.

While configuring, if you specify the group header name (such as SF-Login), then the filter inserts all the subheaders from this group (SF-Login-IdpFedUserId, SF-Login-AuthnContextClassRef, SF-Login-AuthnInstant, SF-Login-ReauthOnOrAfter) in the request headers. In the case where you specify individual subheaders, then the filter inserts only those specified subheaders in the request headers.

A filter logs the high level or debug level information of the requests processed in the log file according to your configurations.

If it is not possible to use these native filters, applications such as Perl or PHP scripts can do the work of these filters. For this purpose, these applications need the following information:

- Cookie name (set by Event plugin - value of `filterSupport.cookieName` property from `tfsconfig.properties`)
- Location of login integration page to redirect if cookie not present
- Directories to be protected
- Unprotected or allowed URLs (like SSO, SLO pages, and so on)
- Passive URLs
- Location of Federation Session Service (FSS)
- `sessionId` as request parameter name for FSS with actual session ID as its value

B Supported Apache Versions

This appendix provides information on supported versions for the Apache filter run-time dependencies on Linux platforms.

Overview

The Apache filter on the Linux platforms includes the following run-time dependencies:

- Apache httpd Web Server
- Apache APR
- cURL/libcURL
- Libxml2
- OpenSSL

The following sections describe the supported versions for each of the above dependencies.

Apache httpd Web Server

The Apache filter module binary is compatible with Apache 2.0.x versions that have a Module Magic Number that starts with 20020903 (has a major version component of). This means that this version of Select Federation is compatible with Apache 2.0.41+ to Apache 2.0.55.

Following is an example of how to determine the server version number and Module Magic Number for your installed version of apache:

```
# /usr/sbin/httpd -V
Server version: Apache/2.0.53
Server built: Sep 5 2005 09:28:47
Server's Module Magic Number: 20020903:9
```

The Apache filter module binary is **not** compatible with Apache 1.3.x.

The Apache filter module was tested successfully with the following versions of Apache httpd web server:

- 2.0.55 on Red Hat Linux AS 3.0
- 2.0.52 on Red Hat Linux AS 4.0 ES Basic Edition

Apache APR

The Apache filter module binary is compatible with Apache APR 0.9.4, 1.2.2 and 1.2.6.

The Apache filter module was tested successfully with the following versions of Apache APR:

- 1.2.2 on Red Hat Linux AS 3.0
- 0.9.4 on Red Hat Linux AS 4.0

cURL/libcURL

The Apache filter module binary is compatible with `cURL` 7.12.1 and 7.15.3.

The Apache filter module was tested successfully with the following versions:

- `cURL` 7.15.3 on Red Hat Linux AS 3.0
- `libcURL` 7.12.1 on Red Hat Linux AS 4.0

Libxml2

The Apache Filter module binary is compatible with `Libxml2 2.6.16` and `2.6.23`.

The Apache Filter module was tested successfully with the following versions of `Libxml2`

- `2.6.23` on Red Hat Linux AS 3.0
- `2.6.16` on Red Hat Linux AS 4.0

OpenSSL

The Apache Filter module binary is compatible with `OpenSSL 0.9.7a` and `0.9.8a`.

The Apache Filter module was tested successfully with the following versions of `OpenSSL`:

- 0.9.8a on Red Hat Linux AS 3.0
- 0.9.7a on Red Hat Linux AS 4.0

C Configuring Server and Client Authentication

The model for setting up the filter for server and client authentication is simple:

- Server authentication requires that you do the following:
  - Enable server authentication.
  - Specify the path to a PEM format server certificate of your SP installation.
- Client authentication requires that you do the following:
  - Enable client authentication.
  - Specify the path to the generated PEM format key file and certificate file, which are used by the filter component in your web server installation.

    To simplify the process, the certificate and key files used by your filter could simply point to the certificate file and key used by your web server.

  - Specify the password for the key file if you have encrypted it.
  - Import whichever Certificate Authority (CA) certificate format (used to sign your filter certificate) is appropriate for your keystore, into your SP's keystore. Your SP's keystore is used for verifying the filter certificate during Client Authentication.

The following section provides a scenario that closely reflects an actual deployment.

Setting Up the Apache Filter

Set up the Apache filter for server and client authentication by completing the following tasks:

- Task 1: Enable Server Authentication
- Task 2: Enable Client Authentication

Task 1: Enable Server Authentication

1. Edit the `<apache_home>/conf/filter.conf` file to enable debug logging for the filter for the duration of the setup:

   ```
   Debug=TRUE
   ```

2. Edit the `<apache_home>/conf/filter.conf` file to enable Server Authentication:

   ```
   SkipServerAuth=FALSE
   ```

3. If your SP certificate is in DER format, convert it to PEM format as follows:

   If you are using a java-based keystore or you generated your certificate using a java-based utility (for example keytool), your certificate may be in DER format.

- Optionally, if your certificate is stored in a java keystore, export it out of the keystore as follows:

  ```
  keytool -export -alias <mycert> -keystore </path/to/keystore>
  -storepass <password> -file </path/to/spcert>
  ```

- Convert the SP certificate from DER to PEM format by entering the following command prompt:

  ```
  openssl x509 -inform DER -in <path/to/spCert> -outform PEM -out <path/
  to/trustedSP.pem>
  ```

4 Provide a path to the newly created `trustedSP.pem` file in the `filter.conf` file:

  ```
  CACerts=/path/to/trustedSP.pem
  ```

  How you provide the path depends on one of the following:

  - If the `trustedSP.pem` file is on the same system as the Apache web server, just provide the path.
  - If the SP and Apache web server are on different machines, first transfer the `trustedSP.pem` file to the system with the Apache web server, and then provide the path.

5 Restart the Apache server and browse to a protected URL to test and check if the server authentication succeeds.

  If you fail to see the protected URL, you can look at the `<apache_home>/logs/filter.conf` file for more details.

When you have enabled and configured server authentication successfully, enable client authentication as described in the next section.

Task 2: Enable Client Authentication

Before you begin, be sure that you have a port configured and listening on your Application Server for performing Client Authentication.

Perform the following steps to enable client authentication:

1 For demonstration purposes, create your own Certificate Authority (CA) as follows:

  Most certificates are signed by a CA.

  a Generate a Certificate Service Request (CSR) for a Certificate Authority (CA) using the `openssl` utility:

  ```
  openssl req -new -newkey -rsa:<key_length> -nodes -out <path/to/ca.csr>
  -keyout <path/to/ca.key>
  ```

  b Since you are your own CA, generate a self-signed CA certificate in PEM format:

  ```
  openssl x509 -trustout -signkey <path/to/ca.key> -days
  <number_of_days_that_the_cert_will_be_valid_for> -outform PEM -req -in
  <path/to/ca.csr> -out ./ca.pem
  ```

  c Optionally, if your deployment needs to make use of a fingerprint for the CA certificate, generate the CA certificate as follows:

  ```
  openssl x509 -fingerprint -in <path/to/ca.pem> -out ./ca.fingerprint
  ```

2 Generate a CSR and private key for the filter component:

```
openssl req -new -newkey -rsa:<filter_cert_key_length> -nodes -out <path/
to/filter.csr> -keyout <path/to/filter.key>
```

3 Create the certificate for the filter by signing the filter's CSR with the CA:

```
openssl x509 -req -days <number_of_days_that_the_cert_will_be_valid_for>
-in <path/to/filter.csr> -CA <path/to/ca.pem> -CAkey <path/to/ca.key>
-CAcreateserial -outform PEM -out ./filter.pem
```

4 Import the CA certificate to a java-based keystore.

a The CA certificate must be in DER format to be imported to a java-based keystore. Therefore, for this scenario, convert the filter certificate from PEM to DER format:

```
openssl x509 -inform PEM -in <path/to/ca.pem> -outform DER -out <path/
to/ca.der>
```

b Import the CA certificate to the file used as the `truststore` for your SP installation.

You can often use the `keytool` utility to do this:

```
keytool -import -alias <CA_Alias> -file <path/to/ca.der> -keystore
<path/to/cacerts_file> -storepass <password>
```

5 Edit the `filter.conf` file to provide the necessary paths and enable client authentication.

You do not need to configure the login URL to use a port that performs client authentication. However, be sure to alter the URL used to query the SP for information, to use the port configured for client authentication.

For example, if you used port 7443 (which does not use client authentication) for the login URL, and port 7444 does use client authentication, you would change the port to 7444 for the URL used to query the SP, as follows:

```
LoginPage=https://sp0.hpovsf.tgx.net:7443/tfs-fs/login.jsp
FederationSessionService=https://sp0.hpovsf.tgx.net:7444/tfs-fs/FSS
SkipClientAuth=FALSE
CertFile=/path/to/filter.pem
KeyFile=/path/to/filter.key
```

You do not need to provide a `passphrase` because the `-nodes` option was enabled to avoid encrypting the key file. Be sure the `passphrase` is commented:

```
#PassPhrase=...
```

6 Restart your SP and the Apache server, and browse to a protected URL to test and check if the client authentication succeeds.

If you fail to see the protected URL, you can look at the `<apache_home>/logs/filter.conf` file for more details.

7 Optionally, if the server and client authentication succeeded, you can turn off debug logging in the `filter.conf` file:

```
Debug=FALSE
```

Be sure to restart the Apache server for the changes to take effect.

D Troubleshooting

To troubleshoot effectively, be sure to configure a log file location, and enable debug logging. For configuration guidelines, see How to Configure the IIS Filter on page 30 and How to Configure the Apache Filter on page 42.

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).

Problems

Problem

The IIS filter configuration interface is not seen after running the installation script.

Solution

First, check that you closed and re-opened the IIS administrative console. Interface changes do not take effect until you do so.

Then, if that did not work, uninstall and re-run the install script.

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 HP OpenView 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 should be set to "Allowed".
- The machine should have been rebooted after modifying the system path.

Index

F

I

J

L

M

O

P

S

T

U