Tivoli® Federated Identity Manager
Version 6.2.1

Configuration Guide

IBM®

GC27-2719-00

Tivoli® Federated Identity Manager
Version 6.2.1

Configuration Guide

IBM®

GC27-2719-00

Note

Before using this information and the product it supports, read the information in "Notices" on page 511.

This edition applies to version 6, release 2, modification 1 of IBM Tivoli Federated Identity Manager (product number 5724-L73) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this publication

IBM® Tivoli® Federated Identity Manager Version 6.2.1 implements solutions for federated single sign-on, Web services security management, and provisioning that are based on open standards. IBM Tivoli Federated Identity Manager extends the authentication and authorization solutions provided by IBM Tivoli Access Manager to simplify the integration of multiple existing Web solutions.

This guide describes how to configure IBM Tivoli Federated Identity Manager.

Intended audience

The target audience for this book includes network security architects, system administrators, network administrators, and system integrators. Readers of this book should have working knowledge of networking security issues, encryption technology, keys, and certificates. Readers should also be familiar with the implementation of authentication and authorization policies in a distributed environment.

This book describes an implementation of a Web services solution that supports multiple Web services standards. Readers should have knowledge of specific Web services standards, as obtained from the documentation produced by the standards body for each respective standard.

Readers should be familiar with the development and deployment of applications for use in a Web services environment. This includes experience with deploying applications into an IBM WebSphere® Application Server environment.

Publications

Read the descriptions of the IBM Tivoli Federated Identity Manager library, the prerequisite publications, and the related publications to determine which publications you might find helpful. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

IBM Tivoli Federated Identity Manager library

The publications in the IBM Tivoli Federated Identity Manager library are:

- *IBM Tivoli Federated Identity Manager Quick Start Guide*

  Provides instructions for getting started with IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager Installation Guide*

  Provides instructions for installing IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager Configuration Guide*

  Provides instructions for configuring IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager for z/OS Program Directory*

  Provides instructions for installing IBM Tivoli Federated Identity Manager on z/OS®.
- *IBM Tivoli Federated Identity Manager Administration Guide*

Provides instructions for completing administration tasks that are required for all deployments.

- *IBM Tivoli Federated Identity Manager Web Services Security Management Guide*

  Provides instructions for completing configuration tasks for Web services security management.
- *IBM Tivoli Federated Identity Manager Auditing Guide*

  Provides instructions for auditing IBM Tivoli Federated Identity Manager events.
- *IBM Tivoli Federated Identity Manager Error Message Reference*

  Provides explanations of the IBM Tivoli Federated Identity Manager error messages.
- *IBM Tivoli Federated Identity Manager Troubleshooting Guide*

  Provides troubleshooting information and instructions for problem solving.

You can obtain the publications from the IBM Tivoli Federated Identity Manager Information Center:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.tivoli.fim.doc_6.2.1/toc.xml

Prerequisite publications

To use the information in this book effectively, you should have some knowledge about related software products, which you can obtain from the following sources:

- IBM Tivoli Access Manager for e-business Information Center:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc/toc.xml
- IBM WebSphere Application Server Version 6.1 Information Center:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

  You can obtain PDF versions of the IBM WebSphere Application Server documentation at:

  http://www.ibm.com/software/webservers/appserv/was/library/

Related publications

You can obtain related publications from the IBM Web sites:

- The IBM Tivoli Federated Identity Manager Business Gateway Information Center at
- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*. This book is available in PDF (Portable Document Format) at http://www.redbooks.ibm.com/redbooks/pdfs/sg246014.pdf or in HTML (Hypertext Markup Language) at http://www.redbooks.ibm.com/redbooks/SG246014/
- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions* (SG24-6394-01). This book is available in PDF at http://www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf or in HTML at http://www.redbooks.ibm.com/redbooks/SG246394/
- The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: http://publib.boulder.ibm.com/tividd/td/tdprodlist.html

- The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available at http://publib.boulder.ibm.com/tividd/glossary/tivoliglossarymst.htm

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Information Center Web site at http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File → Print** window that allows Adobe® Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss.
2. Select your country from the list and click **Go**.
3. Click **About this site** in the main panel to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You also can use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the "Accessibility" topic in the information center at http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.tivoli.fim.doc_6.2.1/toc.xml.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site at http://www.ibm.com/software/tivoli/education.

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online
Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant
The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, see the *IBM Tivoli Federated Identity Manager Installation Guide*. Also see: http://www.ibm.com/software/support/isa.

Troubleshooting Guide
For more information about resolving problems, see the *IBM Tivoli Federated Identity Manager Troubleshooting Guide*.

Conventions used in this book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:**, and **Operating system considerations**:)
- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide: ... where *myname* represents....

`Monospace`

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type

- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX® convention for specifying environment variables and for directory notation.

When using the Windows® command line, replace **$***variable* with **%** *variable***%** for environment variables and replace each forward slash (**/**) with a backslash (****) in directory paths. The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Part 1. Configuration of a domain

The topics in the Configuration section provide a step-by-step guide to configuring a domain. The management console provides wizards to guide you through many of the configuration tasks.

All Tivoli Federated Identity Manager deployments require the deployment of a domain. You must deploy a domain before you configure other features such as single sign-on federation, Web services security management, token services, or User Self Care.

Start with the topic:
- Chapter 1, “Domain configuration,” on page 3

Chapter 1. Domain configuration

A Tivoli Federated Identity Manager domain is a deployment of the Tivoli Federated Identity Manager runtime component to either a WebSphere single server or a WebSphere cluster.

There is one domain per WebSphere cluster. In a single server environment, there can be only one domain.

Each domain is managed independently. You can use installation of the Tivoli Federated Identity Manager management console to manage multiple domains. You can manage only one domain at a time. The domain that is being managed is known as the *active domain*.

When Tivoli Federated Identity Manager is installed, no domains exist. You will use the management console to create a domain. When you installed Tivoli Federated Identity Manager the management service was deployed to a WebSphere server (single server mode) or WebSphere Deployment Manager (WebSphere cluster mode). You will connect with this management service and choose a WebSphere server or cluster to which you will deploy the Tivoli Federated Identity Manager runtime component. When the runtime is deployed and configured, you are ready to configure additional features such as federated single sign-on or Web services security management.

In a WebSphere Network Deployment environment, the deployment and configuration of the Tivoli Federated Identity Manager runtime to cluster members is an automated process. It is not necessary to perform additional installation of Tivoli Federated Identity Manager or Tivoli Access Manager software onto the WebSphere cluster computers. Deployment and configuration of the runtime application to distributed cluster members is performed by the Tivoli Federated Identity Manager management service utilizing the application deployment services of the WebSphere Deployment Manager.

The management console provides a wizard to guide you through the creation of the domain. The following sections list the properties that the wizard prompts you to supply.

Domain management service endpoints properties

Host
The fully qualified domain name for the **Host** where the WebSphere Application Server is running. For example:

```
idp.example.com
```

SOAP Connector Port
The default WebSphere Application Server (standalone) SOAP port is 8880. When you are creating a domain for use with a WebSphere Application Server that is a member of a WebSphere cluster, the SOAP port number might differ. For example, 8879. If you are unsure of the correct SOAP port number, use the WebSphere Application Server administrative console to determine the port.

WebSphere global security properties

WebSphere Application Server can optionally have global security enabled. When global security is enabled, the security properties must be configured for the Tivoli Federated Identity Manager management service. Global security is enabled in most deployments.

Note for z/OS: When deploying on z/OS, WebSphere is typically configured to use a RACF® (or other security product) keyring for certificates. For instructions on setting up certificates for use with Tivoli Federated Identity Manager on z/OS, see the README document on the z/OS distribution media. The instructions describe how to take a certificate from a RACF Keyring, and add it to a Java™ Key Store file for use by Tivoli Federated Identity Manager. The trusted keystore and the optional client keystore files and passwords created by using those instructions should be used instead of the default values (for example, the trust.p12 file) shown below.

Administrative user name
The WebSphere Application Server administrator name. For example, `wsadmin`

Administrative user password
Password for the WebSphere Application Server administrator, as specified during the WebSphere installation.

SSL Trusted Keystore file
Keystore file used by WebSphere Application Server.

When you have installed Tivoli Federated Identity Manager on a computer that uses an existing WebSphere installation, the default path on Linux® or UNIX is:

```
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/etc/trust.p12
```

On Windows:

```
C:\Program Files\IBM\WebSphere\AppServer\
  profiles\AppSrv01\etc\trust.p12
```

When you have installed embedded WebSphere as part of the Tivoli Federated Identity Manager installation, the default path on Linux or UNIX is:

```
/opt/IBM/FIM/ewas/profiles/
 itfimProfile/etc/trust.p12
```

On Windows:

```
C:\Program Files\IBM\FIM\ewas\
   profiles\AppSrv01\etc\trust.p12
```

SSL Trusted Keystore password
The password needed to access the SSL trusted keystore file.

The default password for the WebSphere key is:

```
WebAS
```

SSL Client Keystore file
Keystore file used by WebSphere Application Server.

This keystore file is an optional configuration item. Some WebSphere deployments do not use an SSL Client Keystore file.

SSL Client Keystore password
The password needed to access the SSL client keystore file. This field is needed when you have entered an SSL client keystore file.

WebSphere server or cluster name

The domain wizard prompts for the WebSphere server or cluster name when creating a domain.

Server name
The name of the WebSphere Application Server into which the Tivoli Federated Identity Manager management service will be configured.

The server is a single server, not part of a cluster.

The default name is automatically built by the wizard. For example, on host named `host1`:

```
WebSphere:cell=host1Node01Cell,node=host1Node01,server=server1
```

Cluster name
The name of the WebSphere Application Server cluster into which the Tivoli Federated Identity Manager management service will be configured.

Tivoli Access Manager environment properties

The wizard prompts whether you want to configure into a Tivoli Access Manager environment. Do *not* configure into a Tivoli Access Manager environment if you are using a point of contact server other than WebSEAL. For example, do *not* configure into a Tivoli Access Manager environment if you are using WebSphere as a point of contact server.

The wizard presents the following prompt:

This environment uses Tivoli Access Manager
If you deselect this check box, you do not have to set any properties for Tivoli Access Manager.

If you select this check box, you must specify the properties listed in the following table

Administrator Username
The Tivoli Access Manager administrator. The default ID is sec_master. If you chose an alternate administrator ID when you installed Tivoli Access Manager enter it here.

Administrator Password
The password for the Tivoli Access Manager administrator.

Policy Server Hostname
The fully qualified host name of the computer running the Tivoli Access Manager policy server. For example:

```
idp.example.com
```

Port The port number used to communicate with the policy server. This number matches the port number that you specified when you configured Tivoli Access Manager The Tivoli Access Manager default value is `7135`.

Authorization Server Hostname
The fully qualified host name of the computer running the Tivoli Access Manager authorization server. For example:

```
idp.example.com
```

Port The port number used to communicate with the authorization server. This number matches the port number that you specified when you configured Tivoli Access Manager The Tivoli Access Manager default value is `7136`.

Tivoli Access Manager Domain
The name of the administrative Tivoli Access Manager domain that you specified when you configured Tivoli Access Manager The default value is `Default`.

Worksheet for domain configuration

Complete this worksheet prior to running the wizard to create and deploy the domain and runtime.

The properties on this worksheet are described in Chapter 1, “Domain configuration,” on page 3

Table 1. Domain configuration properties

Property	Your value
Host	
SOAP Connector Port	
Administrative user name	
Administrative user password	
SSL Trusted Keystore file	
SSL Trusted Keystore password	
SSL Client Keystore file	
SSL Client Keystore password	
WebSphere cluster name or WebSphere server name	
This environment uses Tivoli Access Manager	Select or Deselect

When your environment includes Tivoli Access Manager (for example, when using WebSEAL as the point of contact server), you must also supply some Tivoli Access Manager configuration properties

Table 2. Tivoli Access Manager environment properties

Property	Description
Administrator Username	
Administrator Password	
Policy Server Hostname	
Port	
Authorization Server Hostname	
Port	
Tivoli Access Manager Domain	The default value is `Default`.

Creating and deploying a new domain

You must create a domain and deploy a runtime application for each instance of the Tivoli Federated Identity Manager. This task is a prerequisite for configuration of additional Tivoli Federated Identity Manager features such as federated single sign-on or Web services security management. It is also a prerequisite for deployments that use the Tivoli Federated Identity Manager security token service for token exchange. An example of a token exchange scenario is deployment of Tivoli Federated Identity Manager Kerberos constrained delegation with WebSEAL junctions.

Before you begin

A wizard prompts you to supply the necessary configuration properties. You can use the properties on the worksheet that you prepared. For more information on the worksheet, see Chapter 1, "Domain configuration," on page 3

Procedure

1. Verify that the WebSphere Application Server application is running.
2. When you are deploying a domain into a WebSphere Application Server cluster and WebSphere global security is enabled, you must ensure that the WebSphere key files from the Deployment Manager are copied to all nodes in the cluster. Place the keys on each node in the same directory as on the Deployment Manager. WebSphere 6.1 should do this automatically. However, ensure that when the administration console is remote from the Dmgr(Management Service) that the server certificate presented by the DMgr is trusted by the console. One way to do this is to copy the trust store from the DMgr to the console profile.
3. Log in to the WebSphere console and click **Tivoli Federated Identity Manager → Getting Started**.

   The Getting Started portlet is displayed.
4. Click **Manage Domains.** The Domains portlet is displayed
5. Click **Create**. The Domain Wizard displays the Welcome panel.
6. Click **Next**. The Management Service Endpoint panel is displayed.
7. Enter values for the specified properties and click **Next**.
8. The WebSphere Security panel is displayed. Specify whether WebSphere global security is enabled.

   Note: When installing on z/OS, see the README file on the z/OS distribution media for important information about setting WebSphere security properties.
   - When global security is enabled, enter values for the specified properties and click **Next**.
   - When global security is not enabled, leave the remaining properties blank. Click **Next**.
9. Click **Test Connection**. When successful, you will see an information message:

   ```
   FBTCON317I Tivoli Federated Identity Manager connected successfully.
   ```
10. Click **Next**. The WebSphere Target Mapping panel is displayed. Select or enter the name of your server or cluster. When finished, click **Next**.

- When the WebSphere environment consists of a single server, the panel displays a Server name menu with a default name.
- When the WebSphere environment consists of a cluster, the panel displays the Cluster Name menu. This menu lists the names of clusters defined in the cell. Select the name of the cluster to use.

11. The Select Domain panel is displayed. A default name is provided. Accept it or enter a name for the new domain.
12. The Tivoli Access Manager Environment Settings panel is displayed. Select or deselect **This Environment Uses Tivoli Access Manager** as appropriate. and click **Next**. When you select this option, provide values for the rest of the properties.
13. The Summary panel is displayed. Verify that the domain information is correct and click **Finish**.

    The domain is created and the domain wizard exits. The Create Domain Complete panel is displayed.
14. Select both of the check boxes on the Create Domain Complete panel and click **OK**.

    You must complete both of the tasks as part of the initial creation and deployment of the Tivoli Federated Identity Manager management service and runtime:

    - **Make this domain the active management domain**
    - **Open Runtime Node Management upon completion**
15. When you are deploying Tivoli Federated Identity Manager into a WebSphere cluster, ensure that the WebSphere Node Agent is running on all the nodes in the cluster.

    Use the WebSphere administrative console to verify the status of the node agents.
16. The Current® Domain portlet and the Runtime Node Management portlet are displayed. In the Runtime Node Management portlet, click **Deploy Runtime**. A message is displayed:

    ```
    FBTCON355I - A request to deploy the Tivoli Federated Identity Manager
    Runtime is in progress.
    ```

    The following link is displayed:

    ```
    Click to refresh runtime deployment status and check for completion.
    ```

    The Deploy operation may take several minutes. During this time, you can click the link to check for completion. When the deployment is complete, then clicking on the link will return the message:

    ```
    FBTCON132I The Runtime was successfully deployed to the domain.
    ```

    The Runtime Node Management portlet is redrawn. An entry for the runtime is added to the **Runtime Nodes** table for each node in the domain. Also, the **Configure** button is activated.
17. In the Runtime Node table, select the check box for your node and click **Configure**.

    The runtime application is configured into the environment.
18. In a WebSphere cluster environment, configure each node in the cluster by repeating the previous step.
19. When all nodes are configured, click the **Load configuration changes to the Tivoli Federated Identity Runtime** button.

    The button is located in the Current Domain portlet.
20. Continue with the instructions the apply to your deployment:

- In a WebSphere *cluster* environment, continue with "Mapping the runtime to a Web server."
- In a WebSphere *non-clustered* (standalone server) environment, the domain creation and deployment is now complete. Continue with the appropriate instructions for your scenario.

Mapping the runtime to a Web server

About this task

When Tivoli Federated Identity Manager runtime is deployed, it is automatically mapped to the default WebSphere Application Server. In WebSphere cluster environments, WebSphere Application Server is usually deployed into a topology with a Web server such as IBM HTTP Web server. In this case, a WebSphere plug-in has been installed and configured for the IBM HTTP Web server.

The IBM HTTP Web server performs the workload balancing across cluster members. This means that the Tivoli Federated Identity Manager runtime must be mapped to the Web server.

Procedure

1. Log in to the WebSphere administrative console:

   `http://your_host_name:9060/admin`
2. Navigate to **Enterprise Applications -> ITFIM Runtime**.
3. The Configuration tab is displayed. In the Web Module Properties section, select the **Virtual hosts** link. A section titled Apply Multiple Mappings displays a table with a row entry for each Web module.
4. Select the check box for each Web module. Ensure that all check boxes are selected.
5. Accept the default entry of `default_host` in the Virtual host field for each Web module.
6. A message box prompts you to save your changes. Click the **Save** link.
7. The Save panel is displayed. Click the **Save** button.
8. Return to **Enterprise Applications -> ITFIM Runtime**.
9. The Configuration tab is displayed. In the Modules section, select the **Manage Modules** link. The **Enterprise Application -> ITFIM Runtime -> Manage Modules** page is displayed. At the top, you should see the title Manage Modules.
10. Select the check box for *each* of the Web modules. For Tivoli Federated Identity Manager the list of modules can include, but is not limited to:
    - ITFIM-Runtime
    - ITFIM Security Token Service
    - ITFIM Information Service
    - TokenService
    - TrustServerWST13
11. A scrolling window lists Clusters and Servers. Select both of the following entries:
    - The entry for your cluster. For example, `cluster=fimCluster`.
    - The entry for the Web server. For example, `server=webserver1`

12. While both items are highlighted, click **Apply**. In the Module table, the definition for each server and cluster is added to the entry (in the Server column) for each of the Web modules that you selected.
13. Click **OK** at the bottom of the page. A message prompts you to save your changes.
14. Click the **Save** link. The **Enterprise Applications → Save** panel is displayed.
15. Click **Save**.
16. To finish configuring the Tivoli Federated Identity Manager runtime into a WebSphere cluster, continue with "Enabling replication in a WebSphere cluster."

Enabling replication in a WebSphere cluster

About this task

Note: This configuration task applies to WebSphere cluster environments. When you have configured Tivoli Federated Identity Manager runtime to a single server WebSphere environment, skip this topic.

WebSphere supports the use of a *dynamic cache service* for storing application data. The data objects managed by this service can be separated into *cache instances* that can be individually configured. The WebSphere administrator can configure parameters such as cache size, persistence to disk, and others. Each cache instance can belong to a *replication domain* such that the data in the cache is replicated and available to all server that participate in the replication domain.

The Tivoli Federated Identity Manager runtime application uses this capability to enhance performance. When the Tivoli Federated Identity Manager runtime is deployed, some WebSphere configuration steps are automatically performed:

- A replication domain is created. The name of the replication domain is `FIM-`*`your_cluster_name`* or `FIM-`*`your_server_name`*.
- Several cache instances are created that use the replication domain.

Additional configuration is required.

Application servers in the cluster must now have their dynamic cache service configured as a *consumer* of the replication domain.

Note: The steps in this section must be completed for each server in the cluster.

Complete the following steps for *each* application server that is a member of the cluster:

Procedure

1. In the WebSphere administrative console, navigate to **Servers -> Application Servers ->** ***your_server_name*** The properties for the selected server are displayed.
2. In the Container Setting section, expand **Container Services**. Click **Dynamic Cache Service**.
3. In the General Properties section of the screen, go to the Consistency settings section. Select the **Enable cache replication** check box. Verify that the Consistency Settings area has the following values:
   - Full group replication domain

Select the name of the cluster into which you have deployed the runtime application

- Replication type: **Both push and pull**
- Push frequency: **0**

4. Click **OK**. When prompted to save your changes, select the **Save** link. When the next page is displayed, click the **Save** button.
5. in the WebSphere administrative console, navigate to **Servers -> Application Servers ->** *your_server_name*.

   Note: The properties in this section might already be defined.
6. In the **Container Settings** section, select **Session management**.
7. A Configuration tab is displayed. In the Additional Properties section, select **Distributed environment settings**.
8. The General Properties panel is redrawn. Examine the **Distributed environment settings** section.
   a. Select the **Memory-to-memory replication** radio button.
   b. Click the **Memory-to-memory replication** hyperlink.
9. The General Properties panel is displayed.
   a. Set the Replication domain to the name of the cluster into which you have deployed the Tivoli Federated Identity Manager runtime application.
   b. Set Replication mode to **Both client and server**.
10. When prompted to save your changes, select the **Save** link. When the next page is displayed, click the **Save** button.
11. From the Server Cluster panel, select the check box for your cluster and click **Ripplestart**.

    You must restart the cluster to activate the changes you have made.

Part 2. Configuration of a single sign-on federation

The topics in the Configuration section provide a step-by-step guide to configuring a single sign-on federation. The management console provides wizards to guide you through many of the configuration tasks.

Many configuration tasks are common to all federation types. Some configuration tasks are unique to specific federation types.

Complete the configuration tasks in the following order:

1. Review the configuration tasks that are common to all types of federations. Complete the configuration tasks applicable to your deployment.

   Note: Most federation types support a variety of deployment scenarios. The actual steps for each configuration task can vary, depending on the scenario.

   a. Chapter 3, “Identity provider and service provider roles,” on page 17
   b. Chapter 4, “Using keys and certificates to secure communications,” on page 19
   c. Chapter 5, “Configuring LTPA and its keys,” on page 27
   d. Chapter 6, “Setting up message security,” on page 29
   e. Chapter 7, “Setting up transport security,” on page 49
   f. Chapter 8, “Selecting a point of contact server,” on page 61
   g. Chapter 9, “Configuring WebSphere as point of contact server,” on page 65
   h. Chapter 10, “Configuring a Web server plug-in,” on page 93
   i. Chapter 11, “Setting up the alias service database,” on page 107
   j. Chapter 12, “Planning the mapping of user identities,” on page 119
2. Complete the instructions for your federation type:
   - Chapter 13, “SAML federations overview,” on page 137
   - Chapter 18, “Planning an Information Card federation,” on page 213
   - Chapter 21, “OpenID planning overview,” on page 255
   - Chapter 24, “Planning a Liberty federation,” on page 309
   - Chapter 26, “Planning a WS-Federation single sign-on federation,” on page 339

Chapter 2. Overview of configuration tasks for federated single sign-on

Tivoli Federated Identity Manager enables you to establish a single sign-on federation in which users can log in once to access multiple Web applications at different providers.

A federation is a group of two or more trusted business partners that want to initiate or receive the transfer of user identities within the federation. The integrity of the identity is based on existing trust relationships among members of the federation, often codified by a legal agreement. Participation in a federation through federated single sign-on allows a user of one company to seamlessly access resources of that company's federated business partner in a secure and trustworthy manner, usually by using a Web browser.

When you use Tivoli Federated Identity Manager to establish the federation, you can take advantage of the following product features:

- Open standards for single sign-on
- Integration with the single sign-on capabilities of IBM WebSphere Application Server 6.1 eliminating the need for authentication by individual applications
- Support for an unlimited number of federations and the ability to tailor unique configurations for each federation. For example, you can play either an identity provider role or a service provider role in any federation using only one installation of Tivoli Federated Identity Manager.
- Integration support for Web applications running on any of the following types of servers:
  - WebSphere Application Server 5.1 or 6.x
  - Microsoft® Internet Information Server (IIS)
  - IBM HTTP Server (IHS)
  - Apache 2.0 or 2.2 HTTP server
- Simplified Web-based administration

Deployment of a single sign-on federation requires completion of a series of tasks. Some of the tasks are common to all types of federations. Others tasks are specific to the protocol standard for the federation (for example, SAML 2.0).

To deploy a single sign-on federation, you can review the common tasks first and then complete the configuration steps specific to the protocol standard.

Note: You must create a domain before deploying a single sign-on federation. If you have not yet deployed a domain, complete the instructions in Chapter 1, "Domain configuration," on page 3.

The tasks described in the following topics are common to all types of federations. Go through each topic in order before you configure a federation for your selected protocol.

1. Chapter 3, "Identity provider and service provider roles," on page 17
2. Chapter 4, "Using keys and certificates to secure communications," on page 19
3. Chapter 5, "Configuring LTPA and its keys," on page 27

4. Chapter 6, “Setting up message security,” on page 29
5. Chapter 7, “Setting up transport security,” on page 49
6. Chapter 8, “Selecting a point of contact server,” on page 61
7. Chapter 9, “Configuring WebSphere as point of contact server,” on page 65
8. Chapter 10, “Configuring a Web server plug-in,” on page 93
9. Chapter 11, “Setting up the alias service database,” on page 107
10. Chapter 12, “Planning the mapping of user identities,” on page 119

For more information about federation concepts and assistance with architecting a federated identity management solution, refer to the "Federation concepts" chapter of *Enterprise Security Architecture Using IBM Tivoli Security Solutions* redbook at http://www.redbooks.ibm.com/redbooks/pdfs/sg246014.pdf.

Chapter 3. Identity provider and service provider roles

Each partner in a federation has a role. The role is either **Identity Provider** or **Service Provider**.

- **Identity provider**

  An identity provider is a federation partner that vouches for the identity of a user. The Identity Provider authenticates the user and provides an *authentication token* (that is, information that verifies the authenticity of the user) to the service provider.

  The identity provider either directly authenticates the user, such as by validating a user name and password, or indirectly authenticates the user, such as by validating an assertion about the user's identity as presented by a separate identity provider.

  The identity provider handles the management of user identities in order to free the service provider from this responsibility.

- **Service Provider**

  A service provider is a federation partner that provides services to the end user. Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on identity providers to assert the identity of a user, and typically certain attributes about the user that are managed by the identity provider. Service providers may also maintain a local account for the user along with attributes that are unique to their service.

  Service providers can maintain a local account for the user, which can be referenced by an identifier for the user.

  Some federation protocols use different terminology to refer to the service provider role:

  – **Relying party**

    The Information Card protocol specification uses the term Relying Party to describe the service provider role. When you configure the Information Card federation, using the Tivoli Federated Identity Manager wizard, you will choose the Service Provider role for your Relying Party.

  – **Consumer**

    The OpenID protocol specification uses the term Consumer to describe the service provider role. When you configure the OpenID, using the Tivoli Federated Identity Manager wizard, you will choose the Service Provider role for your Consumer.

Before installing Tivoli Federated Identity Manager, you will need to know whether you will be the identity provider or the service provider in each of the federations that you will configure. You will also want to understand the point of contact server options for your role.

Chapter 4. Using keys and certificates to secure communications

In a typical production environment, all messages and the communication of those messages between partners and between users in the federation will be secured. In addition, you might need to secure the communication among the servers in your environment, such as the communication between your server and your user registry.

For example, the SAML standards state that the partners should establish a trust relationship using a Public Key Infrastructure (PKI) and implement Secure Sockets Layer (SSL)-over-HTTP (that is, use HTTPS) to ensure the integrity and confidentiality of messages as they are transported.

Implementing security is a complex topic and is dependent on the configuration of your environment and the security policies of your organization. This overview explains the general concepts of securing the elements in a Tivoli Federated Identity Manager environment. If you need assistance with this topic, review the security recommendations and requirements in the protocol specifications document or contact a computer security consultant.

Message-level security

To secure the content of messages and assertions, the SAML standards specify that public key cryptography be used. Using this method, the federation partners exchange public/private key pairs and use the keys to sign, encrypt, validate and decrypt messages and the assertions within the messages, as required in the SAML standard or as appropriate to their environment.

When you configure a federation in Tivoli Federated Identity Manager, the federation configuration wizard will prompt you with either signing, validation, or encryption *requirements* or signing, validation, or encryption *options* based on your SAML protocol and profile or binding selections. For example, if the choices you make when configuring your federation indicate that a signature is required, the wizard will prompt you for a signing key. If your selections result in an option to sign or not, the wizard will let you make a selection.

Before you use the federation configuration wizard, you must have created the appropriate keys. The information in Chapter 6, “Setting up message security,” on page 29 can help you plan which keys you will need in your environment and provides instructions for creating or obtaining them.

The following sections provide general descriptions of the keys used in SAML federations.

Signing

XML messages and SAML assertions are signed by one partner to protect the integrity of the message. The signature enables the receiving party to check if the message had been changed or modified during transmission. Signing is done using a private key. The partner who receives the signed XML message or SAML assertion will need the X.509 certificate (public key) that corresponds to the message signer's private key. By default, the X.509 certificate (public key) is included with the signature as a base64-encoded X.509 certificate. However, you have the option of specifying which certificate data you want to include with your signatures.

Validation

The signatures in messages and assertions can be validated by the partner who receives them. Validation confirms that the signer's identity as been assured. Validation is done using the public key of the partner who signed the messages or assertions.

Encryption and decryption

In SAML 2.0, messages can be encrypted in addition to being signed. The use of the public/private key pair during encryption and decryption differs from its use during signing and validation. Encryption is done using the *public key of the intended recipient*. In other words, for one partner to encrypt a message, that partner must have the public key of the partner to whom the message is being sent. The partner who receives the encrypted message must use its *private* key to decrypt the message. In Tivoli Federation Identity Manager, when SAML 2.0 is used, both partners must obtain their own public/private key pairs to be used for encryption. They must then exchange their public keys so that each partner can encrypt messages to the other.

Transport-level security

Message-level security, as described in the preceding section, protects only the content of the message. To protect the message as it is communicated (transported) between the partners, SAML recommends using Secure Sockets Layer (SSL) with server authentication and in some cases with mutual authentication.

SSL is a protocol that establishes authenticity, integrity, and confidentiality among parties who are transmitting data over various other protocols (such as HTTP) in a network.

Note: SSL is a complex topic. This overview is simplified and brief and serves as only an introduction so that you will be familiar with the basic concepts and terminology used in this book.

Server authentication

In a Tivoli Federated Identity Manager environment, SSL is used to protect the endpoints at which SAML messages are sent and received. In an SSL-protected communication between federation partners, one partner acts as *client* (that is, the party who is requesting data) and the other partner acts as the *server* (that is, the receiver of the request and the responder to the request).

In a SAML 1.x federation, a single sign-on request is received at the identity provider partner. Therefore, when an SSL connection is established between the federation partners, the identity provider partner acts as the *server* and the service provider partner acts as the *client*.

In a SAML 2.0 federation, a single sign-on request can be received by either partner. Therefore, either partner could be the server or client.

SSL can be configured on the server only (*server authentication)* or on both the server and the client (*mutual authentication*). The SAML standards recommend that, at a minimum, server authentication be used between partners. The addition of mutual authentication provides added security.

To enable server authentication, you will need to create a public/private key pair and obtain a certificate, which is used by your server to authenticate itself to the client. The certificate is referred to as a *server certificate* or a *personal certificate*.

Although you can create your own server certificate (using software that supports certificate creation), in a production environment, you will generally want to obtain a server certificate from a third-party, referred to as a *certificate authority* or *CA*, that issues certificates. Prior to attempting an SSL connection, the client that the server will present its certificate to must obtain the certificate of the CA that issued the server certificate. The client maintains a list of trusted issuers and will add the CA certificate to that list.

The server certificate contains information such as the server certificate public key, the certificate serial number, the certificate validity period, the server's distinguished name (which includes the host name associated with the server), the issuer's distinguished name, and the issuer's digital signature.

To establish the SSL connection, the server presents its certificate and the client must verify it. For example, the client checks its list of trusted issuers (certificate authorities) to see if the server's certificate issuer is trusted and it compares the issuer's digital signature in the server certificate to the issuer's digital signature in the CA certificate.

The server must export its CA certificate and provide it to its client partner.

In summary, server authentication requires the following keys and certificates:

Table 3. SSL server authentication certificate requirements

Certificate required	**Who must obtain certificate**	**Notes®**
Server certificate and private key associated with that certificate	Partner acting as the server	In a SAML 1.x federation, the identity provider will always act as the server.
CA certificate of the server certificate issuer	Partner acting as the client	In a SAML 1.x federation, the service provider will always act as the client.

Instructions for enabling SSL are described in “Enabling SSL on the WebSphere Application Server” on page 49.

Client authentication

A server can be configured to require authentication from its clients in order to confirm their identities. Tivoli Federated Identity Manager accepts either of the following client authentication methods:

Basic (password-based) authentication
If basic authentication is configured, the server requests the client to supply a username and password to authenticate. No certificates are used with this method.

Client certificate authentication

A *client certificate* is similar to a server certificate. To obtain a client certificate, the client will typically request it from a CA. Prior to establishing a federation, the partners will typically agree on a CA to use for signing the client certificate. The server must ensure that CA is in its list of trusted issuers. When client certificate authentication is configured, the server requests authentication from the client and the client responds by sending its client certificate and its digital signature in a randomly generated piece of data to the server. The client certificate generally includes the client's public key, the certificate's serial number, the certificate's validity period, the client's distinguished name, the issuer's distingushed name, and the issuer's digital signature. The server verifies the client information. For example, the client exports its certificate and provides it to the server partner. Then the server uses the client's public key in the client certificate to validate the client's digital signature, checks its list of trusted issuers (Certificate Authorities) to see if the client certificate issuer is trusted and compares the issuer's digital signature in the client certificate to the issuer's digital signature in the CA certificate.

If client certificate authentication is used, the following certificates are required:

Table 4. SSL client authentication certificate requirements

Certificate required	Who must obtain certificate	Notes
Client certificate and its associated private key	Partner acting as the client	In a SAML 1.x federation, the service provider will always act as the client.
CA certificate of the client certificate issuer	Partner acting as the server	In a SAML 1.x federation, the identity provider will always act as the server.

Partners acting as *servers* will need to follow instructions for configuring a client authentication requirement on their servers, “Configuring client authentication requirements” on page 54.

Partners acting as *clients whose partners require client certificate authentication* will need to follow instructions for configuring their client certificates, “Configuring your client certificates” on page 58.

Storage and management of keys and certificates

Keys and certificates are stored in keystores and truststores.

Keystores

Private keys and personal certificates are stored in keystores.

Truststores

Public keys and CA certificates are stored in truststores. A truststore is a keystore that by convention contains only trusted keys and certificates.

In your Tivoli Federated Identity Manager environment, some keys and certificates are stored in the WebSphere Application Server keystores and truststores and some are stored in the Tivoli Federated Identity Manager keystores and truststores and are managed by a Tivoli Federated Identity Manager function called the *key service*. The location depends on the purpose of the keys and certificates being used.

Keys and certificates stored in a WebSphere Application Server keystore and truststore:

- SSL server certificates and their private keys (in the WebSphere keystore of the server partner)
- CA certificate for clients that will present a client certificate (in the WebSphere truststore of the server partner)

Keys and certificates stored in a Tivoli Federated Identity Manager keystore and truststore:

- SSL client certificates (those used for client certificate authentication) and their private keys (in the keystore of the client)
- CA certificates for servers that have SSL server authentication configured (in the truststore of the client)
- Signing keys, validation keys and encryption keys are also managed in the keystores and truststores in Tivoli Federated Identity Manager. For example:

  Signing keys
  These are private keys that are stored in the keystores.

  Validation keys
  These are public keys that correspond to the private keys used for signing. These are stored in the truststores.

  Encryption keys
  - The key used to encrypt data is a public key that was obtained from your partner. You would store it in your truststore.
  - The key used to decrypt data is a private key. You would store it in your keystore.

By default, both WebSphere Application Server and Tivoli Federated Identity Manager have keystores, truststores, keys, and certificates that are intended to be used in test environments.

WebSphere Application Server

During profile creation, WebSphere Application Server creates:

- key.p12 keystore
- trust.p12 truststore
- a default self-signed certificate in the key.p12 keystore

The password for both the keystore and truststore is `WebAS`.

Tivoli Federated Identity Manager

Tivoli Federated Identity Manager supplies two default Java keystores, a self-signed certificate, and some CA certificates:

- DefaultKeyStore.jks for signing and encryption keys. (Private keys)
- DefaultTrustedKeyStore.jks for CA certificates
- A self-signed certificate, with the alias `testkey` which can be used as a signing key in a test environment in the keystore
- Several CA certificates in the truststore

The default password for the keystores is `testonly`.

Because these default keys and certificates are for test purposes only, you will need to create new keys and certificates, and you might also want to create new keystores when you configure Tivoli Federated Identity Manager.

For more information, see “Creation of keystores, keys, and certificates.”

Creation of keystores, keys, and certificates

As described in the preceding sections, to configure message-level security and transport-level security you will use public/private key pairs and certificates. To use the appropriate keys and certificates, you will need to follow a general process for creating them and for creating the keystores where you will store them, if you choose not to use the default keystores.

The general steps for creating keys and certificates and their keystores are:

1. Create the keystore (either a keystore or truststore) or use an existing one.
2. Create the certificate request, which generates a public/private key pair and can be sent to a certificate authority (CA). The certificate request contains the public key and data about you (the requestor) of the certificate.
3. Send the certificate request to the CA. The CA issues the certificate.
4. Receive the certificate from the CA and import it into the appropriate keystores.
5. Share the public keys of the certificates with your partner as needed.

In addition, you will also need to import some keys and certificates from your partner into your keystores.

Both WebSphere Application Server and Tivoli Federated Identity Manager provide utilities for creating certificate requests and for receiving the request from the certificate authority.

Information about completing all message-level security tasks and transport-level security tasks, including the creation of keystores, keys, and certificates using the utilities is in the following sections of this book:

Message-level security instructions:
Chapter 6, “Setting up message security,” on page 29.

Transport-level security instructions:
Chapter 7, “Setting up transport security,” on page 49

Key selection criteria

Configure the order of certificates or keys by using the runtime key selection criteria.

By default, Tivoli Federated Identity Manager, version 6.1, builds a list of certificates or keys sharing the same SubjectDN and optimized from longest to shortest lifetime. This product function, known as Auto Key Rollover, has the following characteristics:

- When signing documents, the function uses a valid key with the shortest remaining lifetime (for example, the oldest X.509 Certificate or Private Key).
- During validation, the function cycles through the list of keys for the given SubjectDN until validation is successful. An unsuccessful validation means that all the available keys were invalid.

By using the runtime key selection criteria, however, you can configure the order of certificates or keys in the following ways:

- Alias only: The selected key only, without Auto rollover. If the key is invalid, the software indicates failure.
- Shortest lifetime: For signing, a valid key with the shortest available lifetime. For validation, key lifetime availability runs from shortest to longest.
- Longest Lifetime: For signing, a valid key with the longest available lifetime. For validation, key lifetime availability runs from longest to shortest.

Chapter 5. Configuring LTPA and its keys

You must review the Lightweight Third Party Authentication (LTPA) on your WebSphere Application Server after you have installed Tivoli Federated Identity Manager. You can choose to use the default LTPA configuration or modify the configuration so that it is appropriate for your environment.

About this task

The default LTPA configuration is as follows:

Key set group
The LTPA keys are used to encrypt and decrypt data that is sent between the servers. The keys are stored in sets and the sets are stored in groups. The default key set group is NodeLTPAKeySetGroup.

Key sets
The default key sets are NodeLTPAKeyPair and NodeLTPASecret.

Key generation
By default, LTPA keys are automatically generated the first time you start the server after installation and will be automatically regenerated every 12 weeks at 2200 hours (on a 24-hour clock) on Sundays.

Attention: If you are using a separate target application server in your configuration, such as a separate WebSphere Application Server or a server that is supported by the Tivoli Federated Identity Manager Web server plug-in, the LTPA keys must be on your WebSphere Application Server point of contact server and on your target application server. If you automatically generate keys, you must make sure that you keep the keys on the application server in sync with the keys that are generated on your WebSphere Application Server point of contact server. For more information about exporting keys from your WebSphere Application Server and importing them to your application server, refer to “Exporting LTPA key from the point of contact server” on page 91 and either “Importing the LTPA key to the WebSphere Application Server” on page 97 or “Copying the LTPA key to the Web server” on page 100.

Authentication cache timeout
This value specifies how long an LTPA token is valid in minutes. The default time is 10 minutes.

Timeout value for forwarded credentials between servers
This value specifies how long the server credentials from another server are valid before they expire. The default value is 120 minutes.

To review or modify these settings:

Procedure

1. Log in to the console.
2. Click **Security** → **Secure administration, applications, and infrastructure**.

   Secure administration, applications, and infrastructure panel is displayed.

3. On the left, click **Authentication mechanisms and expiration**. The Configuration tab is displayed. Use this tab to review or modify the Key set group defined, the authentication cache timeout and the timeout value for forwarded credentials between servers.
4. To modify the key set group and key generation settings, click **Key set groups**. Make changes as appropriate for your environment, and then click **Apply**. Return to the previous Configuration tab.
5. In the Authentication expiration section of the Configuration tab, review or modify the values in the **Authentication cache timeout** field and the **Timeout value for forwarded credentials between servers** field. Click **Apply** when you are done.
6. Save the changes to the master configuration file as prompted.

What to do next

Continue with the configuration of your environment. For example, continue with Chapter 6, "Setting up message security," on page 29.

Chapter 6. Setting up message security

Tivoli Federated Identity Manager uses certificates (pairs of public and private keys) to secure messages.

Before establishing a federation, you and your partner must decide what security configurations you will use within your federation. Then, you will need to create or request your certificates or obtain them from your partner, as appropriate, and import them into the Tivoli Federated Identity Manager key service.

Note: Instructions for configuring SSL-related certificates, such as server certificates, client certificates, and client authentication requirements are described in Chapter 7, "Setting up transport security," on page 49. Except for the topics related to preparing your keystores, the topics in this chapter cover only message-level security.

Use the following tasks to set up message security in your environment:

1. Prepare your keystores. See "Preparing the keystores."
2. Discuss message security requirements with your partner and make a list of the keystores and certificates that each of you will need. Consider using the checklists in "Planning message-level security" on page 32.
3. Obtain the necessary certificates for your environment. See "Obtaining your keys and certificates" on page 34.
4. Add your certificates into your keystores. See "Adding your certificates to your keystore" on page 37.
5. Obtain any certificates you need from your partner. See "Obtaining a certificate from your partner" on page 40.
6. Provide your partner with any of your certificates that might be needed by that partner. See "Providing certificates to your partner" on page 42.
7. If any of the certificates you will use are PKCS#12 files, you will need to update your Java cryptography policy. See "Updating the cryptography policy" on page 44.
8. If you are setting up a production environment and will not use the default keystores and certificates, you might want to remove them so that they are not used inadvertently. See "Removing default keystores" on page 45.

Preparing the keystores

Regardless of the role you will play in a federation or the SAML standard you are using, you must prepare keystores in the Tivoli Federated Identity Manager key service. The keystores will store keys and certificates that are used to secure the content and transport of messages.

About this task

You will have at least two keystores in the key service:

Signing/Encryption keystore
The keystore is where you store your private keys (those you use for signing and decryption, and for your client certificate, if you will be a

client in an SSL connection with your partner and that partner requires that you authenticate with a certificate).

CA Certificates keystore (called a truststore or trusted keystore)
This keystore is where you store your partner's public keys (which you will use to validate signatures or encrypt data to your partner) and CA certificates for the CAs that you trust.

To prepare the keystore and truststore for your environment, you can either:

- Use the default keystore and truststore and change their passwords, so that their default password is no longer used, as described in “Changing a keystore password.”
- Create a new keystore and a new truststore, as described in “Creating a keystore” on page 31 and then import them into the key service.

You can create as many keystores and truststores as you want to make it easier to categorize keys that are unique to your federations.

Changing a keystore password

You can change the password of a keystore or truststore using the console.

About this task

You might want to change the keystore passwords in any of the following situations:

- You want to use the default keystore or truststore in a production environment.
- The keystore password has been compromised.
- Your security policy requires that the keystore passwords be changed at a regular interval.

Procedure

1. Log in to the console.
2. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
3. Select a keystore from the Keystore table. The **Change Password** button is activated.
4. Click **Change Password**. The Change Keystore Password panel is displayed.
5. Enter the original password and the new password. The original password for the default keystore and truststore is `testonly`.
6. Click **OK**. The password is changed.
7. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

What to do next

Repeat the process for each keystore that has a password that must be changed. Then continue with either creating new keystores or planning your message-level security.

Creating a keystore

You might need to create a keystore if you need additional keystores or if you do not want to use the default keystores. **Note:** The Tivoli Federated Identity Manager key service supports only Java keystores (.jks)

About this task

Tivoli Federated Identity Manager does not provide a utility for creating keystores. However, you can use any of several key generation utilities to create a keystore. For example, use the keytool utility that is included with WebSphere Application Server to create a keystore file as follows:

```
keytool -import -noprompt -trustcacerts -alias myca
-file myca.pem -keystore mykeys.jks -storepass passw0rd
```

For details about the keytool utility, refer to the WebSphere Application Server 6.1 Information Center http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

What to do next

You must import the keystore into the Tivoli Federated Identity Manager key service. See “Importing a keystore” for details.

Importing a keystore

If you have created a keystore, you must import it into the Tivoli Federated Identity Manager key service before you can use it.

About this task

Procedure

1. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
2. Click **Import**. The Import Wizard starts and displays the Import Keystore panel.
3. Enter a fully qualified path in the **Location of keystore file** field. For example:

   ```
   /tmp/mykeys.jks
   ```

   Optionally, you can click **Browse** to find the keystore file on the file system.

   Note: The keystore to be imported must be on the same machine as the browser being used to access the administration console.
4. Enter the **Keystore Password**.

   Attention: Private (personal) keys in a keystore can be encrypted with a password. The keystore itself is also protected by a password. However, the key service keeps only one password for a keystore. Therefore, an encrypted private key and its keystore must have the same password.
5. Enter the **Keystore Name**.
6. Specify the type.
   - **Signing/Encryption Keys**
   - **CA Certificates**

   The type indicates the type of key or certificate you want to store in the keystore. For example, if you want to use this keystore to store certificates from your partner, you would choose CA Certificates. If you want to use the keystore to store your own signing keys, you would choose Signing/Encryption Keys.

The type is for information purposes only, and does not prevent you from adding other key types to the keystore. However, using the types consistently (one for private and one for public) can help you organize and locate keys more easily.

7. Repeat these steps for each keystore you need to create for your certificates and your partner's certificates.
8. Click **Finish**.

What to do next

Your keystore is ready to receive keys and certificates. Repeat to import other keystores or continue with “Planning message-level security.”

Planning message-level security

To begin the process of setting up message security for your environment, you will need to determine your requirements.

Meet with your partner and discuss your environments. Use the following checklist tables during your discussion. Consider recording your requirements in the checklist tables.

The options you have for securing the content of messages depend on the SAML standard and profile you are using in your federation and sometimes the role (identity provider or service provider) you have in the federation.

In general, you will have the option to sign messages, sign assertions, and validate your partner's signatures. In a SAML 2.0 federation, each partner must also encrypt data they send to each other and then decrypt the data so it can be used in the federation.

- To sign, you will use your private key from a public/private key pair.
- To validate, you will use your partner's public key that corresponds to the private key the partner used to sign the data.
- To encrypt, you will use your partner's public key that corresponds to the private key the partner uses to decrypt the data. Likewise, you will give your public key to your partner, who will use it to encrypt data to you, and then you will decrypt that data using your corresponding private key.

Use the following checklist to identify which public/private key pairs you need and which keys you need to exchange with your partner.

Your keys

You will use the *private key* of a public/private key pair to perform the actions listed in the following table. You can use the same key for all of these actions or you can use different keys for each action. All of the keys are optional and available to all SAML standards and provider roles *unless* otherwise noted in the Notes column.

Table 5. Your keys

Purpose of the key	Alias of public/private key pair	Keystore in which to store key	Notes
Signing key for messages			Required if you are an identity provider in a SAML 1.x federation. **Note:** In SAML 2.0, the same key is used for signing messages and assertions.
Signing key for assertions			Required for identity providers. **Note:** In SAML 2.0, the same key is used for signing messages and assertions.
Decryption key			Required in SAML 2.0. Not available in SAML 1.x federation.

Keys you need from your partner

You will use the *public key* from your partner's public/private key pair to perform the actions listed. The Notes column indicates if a key is required or if it cannot be used because of a specific provider role or the SAML specification used by the federation. In most cases, you will obtain these keys from your partner by way of a metadata file. However, if you are using a SAML 1.x federation, you might need to obtain these keys manually. Consider sharing this table with your partner to ensure that your partner knows what keys it must provide to you.

Table 6. Keys you need from your partner

Purpose of the key	Alias of public key	Truststore in which to store key	Notes
Validation key for message signatures			Corresponds to your partner's signing key. Required if your partner signs messages.
Validation key for assertion signatures			Corresponds to your partner's signing key. Required if your partner signs assertions. Not available for identity providers using SAML 1.x

Table 6. Keys you need from your partner (continued)

Purpose of the key	Alias of public key	Truststore in which to store key	Notes
Encryption key			Corresponds to your partner's decryption key. Required in SAML 2.0 Not available in a SAML 1.x federation

Keys you must provide to your partner

You will *provide your public key* from your public/private key pair to your partner so that your partner can perform the actions listed. The Notes column indicates if a key is required or if it cannot be used because of a specific provider role or the SAML specification used by the federation. In most cases, you will provide these keys by exporting your federation properties into a metadata file that your partner will import into its configuration. However, if you are using a SAML 1.x federation, you might need to export these keys from your federation and provide them to your partner manually.

Table 7. Keys you must provide to your partner

Purpose of the key	Alias of public/private key pair	Keystore in which key is stored	Notes
Validation key for message signatures			Corresponds to your signing key. Required if you sign messages.
Validation key for assertion signatures			Corresponds to your signing key. Required if you sign assertions. Not available for identity providers using SAML 1.x
Encryption key			Corresponds to your decryption key. Required in SAML 2.0 Not available in a SAML 1.x federation

Obtaining your keys and certificates

After you have determined which keys and certificates you need for signing and decryption, you must obtain them.

About this task

In general, the private keys you will need to obtain include:

Signing key
: If you will sign messages or assertions, you must have public/private key pair and use the private key for signing.

Decryption key
: If you are using SAML 2.0, your partner must encrypt data to you. You must have a public/private key pair for this purpose. Your partner will use your public key to encrypt data it sends to you and you will use your private key to decrypt it.

The method you use to obtain these keys depends on whether you are using a test environment or a production environment:

- In a test environment, you could use the default testkey or create a self-signed certificate. See “Using the default key as your signing and decryption key” or “Creating self-signed certificates.”
- In a production environment, you would want to request your keys from a certificate authority. See “Requesting CA-signed certificates” on page 36.

The following types of certificates can be used in the Tivoli Federated Identity Manager key service. When you obtain certificates, be sure to use these supported types:

- PEM

  Privacy-Enhanced Message. These are public certificates in PEM format.
- PKCS#12

  Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard.

  Before using PKCS#12 certificates, you will need to update the cryptography policy. See “Updating the cryptography policy” on page 44.

Using the default key as your signing and decryption key

In a test environment, you can use the testkey that is in the DefaultKeyStore as your signing and decryption key.

About this task

Ensure that the `testkey` is in the keystore. No additional preparation is needed to use this key.

Creating self-signed certificates

In a test environment, you could use a self-signed certificate for your signing and decryption key or for the client authentication certificate you might be required to present to the server during an SSL communication.

About this task

A self-signed certificate is a public/private key pair that is randomly generated and is signed by its own private key. You can create a self-signed certificate using the utility in Tivoli Federated Identity Manager or using another key creation utility. The following procedure describes using the Tivoli Federated Identity Manager utility.

Note: This procedure is supported only on WebSphere Application Server Version 6.1 installations.

Procedure

1. Log in to the console.
2. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
3. Select a keystore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel is displayed.
5. Type your keystore password and click **OK**.
6. Click **Create Self-Signed Certificate**. The Create Self-Signed Certificate panel is displayed.
7. Complete the fields. Then click **OK**. A public/private key pair is added to the keystore.
8. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

To verify that the certificate was created, repeat steps 1 through 5.

Requesting CA-signed certificates

In a production environment, you will want to obtain your certificates for signing, decryption, and client authentication from a certificate authority that will sign the certificates. You can generate a certificate sign request using the console.

Before you begin

Ensure that you have a keystore ready in which to store the certificate request, and later, the certificate.

About this task

A certificate sign request (CSR) is an electronic file that can be sent (using e-mail, FTP, or other communication methods as required by the certificate authority) to a certificate authority (a CA, such as VeriSign, Thawte, and so on) as a request for a certificate that is signed by that CA.

The CA will use the data contained within the CSR and will generate the certificate and then sign the certificate with its own private key.

The signature of the CA validates the certificate as being trustworthy.

A CSR contains the following data:

- The identity of the requestor (you) in the form of a subject distinguished name
- The extensions for the certificate (if any)
- The public key for the certificate
- The algorithms to be used for the signature and the key

When the request is generated, a temporary self-signed certificate is created in the keystore. This temporary certificate is replaced by the CA-signed certificate when you receive it from the CA.

Note: This procedure is supported only on WebSphere Application Server Version 6.1 installations.

Procedure

1. Log in to the console.
2. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
3. Select a keystore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel is displayed.
5. Type your keystore password and click **OK**.
6. Click **Create Certificate Request**. The Create a certificate request panel is displayed.
7. Complete the fields. Then click **OK**. The Generated Certificate Signature Request window is displayed.
8. Copy and paste the request text into a text file or click the **Export Certificate Signature Request** button to download it. The file that you save or download is ready for you to send to a CA.
9. Click **Done** when you have saved the file. A public/private key pair is added to the keystore and a file with the encoded BASE64 data is created. The temporary self-signed certificate will need to be replaced with the signed certificate from the CA.
10. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

Repeat these steps for each certificate you want to request. For example, you might want a separate certificate for each activity (such as signing, decryption, and client authentication) or you might want to use one certificate for all activities. When you have created all of your certificate sign requests, follow your CA's instructions for transmitting the request file. Then, continue with the steps for receiving a CA certificate from the CA in “Receiving a signed certificate from a CA” on page 39.

Adding your certificates to your keystore

Before you establish your federation, you must add the keys you will use for signing and decryption to your keystore.

About this task

The method by which you add your keys to your keystore depends on the way in which you obtained your keys:

Created a self-signed certificate

If you created a self-signed certificate using the utility in Tivoli Federated Identity Manager, the certificate was automatically imported into your keystore. If you created a self-signed certificate but used a utility other than the one provided with Tivoli Federated Identity Manager, you will need to import your certificate into the keystore as described in “Importing a certificate” on page 38.

Requested a signed certificate

If you generated a certificate sign request and sent that request to a CA, you will receive the certificate into your keystore as described in "Receiving a signed certificate from a CA" on page 39.

Importing a certificate

About this task

You might need to import a certificate if you received the certificate in either of the following ways:

- You created a self-signed certificate using a utility other than the one provided with Tivoli Federated Identity Manager
- You manually obtained a certificate from a CA

You might also need to import a certificate that you have received from your partner. For more information on importing partner certificates, see "Importing a certificate from your partner" on page 41.

Attention: Private (personal) keys in a keystore can be encrypted with a password. The keystore itself is also protected by a password. However, the key service keeps only one password for a keystore. Therefore, an encrypted private key and its keystore must have the same password.

Use this task to import either:

- A certificate from a PEM file
- A key from a PKCS#12 file

  Note: If you will use a PKCS#12 file, be sure to also follow the instructions in "Updating the cryptography policy" on page 44.

Ensure that your key or certificate is ready and available before continuing with this procedure.

Imported keys are enabled by default.

Procedure

1. Click **Tivoli Federated Identity Manager → Key Service**.

   The Keystores panel is displayed.
2. Select a keystore from the Keystore table to store your public/private key pair. The **View Keys** button is activated.

Attention: Do not import private keys (such as signing keys or encryption keys) into a **CA Certificate** keystore. The CA Certificate type of keystores do not store a key password, which is required for private keys.

3. Click **View Keys**. Then enter the keystore password when prompted and click **OK**. The Keys panel is displayed. Keys in the selected keystore are listed.
4. Click the **Import** button. The Key Wizard starts and displays the Welcome panel.
5. Click **Next** The Keystore Format panel is displayed.
6. Select the appropriate **Keystore format** for the file you want to import. Then, click **Next**. The formats are:

   PEM)

   (Privacy-Enhanced Message) Public certificate

PKCS#12
Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard

JKS
Java Key Store

The **Upload Key File** panel is displayed.

7. Specify the path to the location of the key, and if prompted, a password for the key file. Then click **Next**.
8. Specify a label for the key and, if prompted, select the key to import. Then click **Next**.
9. A summary panel is displayed. Click **Finish** to exit the wizard.
10. Repeat these steps to import all the keys and certificates you will use in the federation.

What to do next

Next you will want to add your partner's keys into your truststore. See "Obtaining a certificate from your partner" on page 40.

Receiving a signed certificate from a CA

If you created a certificate sign request using the console and sent it to a CA, you can receive the certificate from the CA to your keystore.

Before you begin

Ensure that you have completed the steps in "Requesting CA-signed certificates" on page 36 and have saved the certificate from the CA to a location that is accessible to the key service.

Procedure

1. Log in to the console.
2. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
3. Select the keystore where the CSR was generated in the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel is displayed.
5. Type your keystore password and click **OK**.
6. Click **Receive Certificate from CA**.
7. Select the location of the certificate that you received from the CA. Then click **OK**. The temporary self-signed certificate in the keystore will be replaced with the received signed certificate.
8. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

Next you will want to add your partner's keys into your truststore. See "Obtaining a certificate from your partner" on page 40.

Obtaining a certificate from your partner

Depending on the requirements of your environment, you might need to obtain certificates from your partner.

Before you begin

Use the worksheet, “Planning message-level security” on page 32, to determine which certificates you might need from your partner. In general, the public keys you will need to obtain from your partner include:

Validation key
If you partner signs messages or assertions and you will validate those signatures, you must have the public key that corresponds to the key that was used sign.

Encryption key
If you must encrypt data that you send to your partner, you must obtain a public key from your partner. You will use the public key to encrypt and your partner will use its corresponding private key to decrypt.

About this task

Usually in a SAML 2. 0 federation, you will receive your partner's validation and encryption keys in a metadata file from your partner. This process is further explained in “Importing certificates from your partner's metadata file.” In addition to the keys, other information from your partner, such as company name, and so on, is included in the metadata file. When you create your partner in the federation, you will be prompted to save the partner's keys into the appropriate keystore. Partner keys should be saved to your truststore.

If you have already received your partner's metadata and only need to receive a new certificate from your partner or if you are using a SAML 1.x federation, you can manually receive the keys (such as through an email, FTP, or other media) and then import them into your truststore using the instructions in “Importing a certificate from your partner” on page 41.

Importing certificates from your partner's metadata file

If your partner will be supplying you with a metadata file of its federation configuration, your partner's public keys should be part of that file.

About this task

Depending on the message-level security and the SAML specification that you and your partner are using in the federation, the metadata file should include one or more of the following public keys:

- Key for validating signed assertions, if the partner signs assertions and you will validate them
- Key for validating signed messages, if the partner signs messages and you will validate them
- Key for encrypting (in a SAML 2.0 federation)

Refer to “Planning message-level security” on page 32.

If your partner is using Tivoli Federated Identity Manager, the public keys that correspond to the private keys that the partner defined in its configuration are automatically added to the metadata file when the partner exports its configuration.

If you are obtaining your partner's keys from a metadata file, you will want to import the metadata as part of establishing your federation. To continue, complete the remaining tasks in this chapter.

Importing a certificate from your partner

You can obtain your partner's public keys in several ways, including from an SSL connection or by importing a metadata file of your partner's configuration. However, if either of these methods are not available, you can obtain the keys manually and import them.

Before you begin

Before beginning this task, ensure that you have received one or more public keys from your partner (such as over FTP, through e-mail, or another transfer method).

About this task

You might need to import a certificate in any of the following situations:

- A self-signed certificate that you created using a utility other than the one provided with Tivoli Federated Identity Manager
- Certificate obtained manually from a CA

You might also need to import a certificate that you have received from your partner. For more information on importing partner certificates, see “Importing a certificate from your partner.”

Attention: Private (personal) keys in a keystore can be encrypted with a password. The keystore itself is also protected by a password. However, the key service keeps only one password for a keystore. Therefore, an encrypted private key and its keystore must have the same password.

Use this task to import either:

- A certificate from a PEM file
- A key from a PKCS#12 file

  Note: If you will use a PKCS#12 file, be sure to also follow the instructions in “Updating the cryptography policy” on page 44.

Ensure that your key or certificate is ready and available before continuing with this procedure.

Imported keys are enabled by default.

Procedure

1. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
2. Select a keystore from the Keystore table to store your public/private key pair. The **View Keys** button is activated.

Attention: Do not import private keys (such as signing keys or encryption keys) into a **CA Certificate** keystore. The CA Certificate type of keystores do not store a key password, which is required for private keys.

3. Click **View Keys**. Then enter the keystore password when prompted and click **OK**. The Keys panel is displayed. Keys in the selected keystore are listed.
4. Click the **Import** button. The Key Wizard starts and displays theWelcome panel.
5. Click **Next** The Keystore Format panel is displayed.
6. Select the appropriate **Keystore format** for the file you want to import. Then, click **Next**. The formats are:

   PEM)
   (Privacy-Enhanced Message) Public certificate

   PKCS#12
   Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard

   JKS
   Java Key Store

   The **Upload Key File** panel is displayed.
7. Specify the path to the location of the key, and if prompted, a password for the key file. Then click **Next**.
8. Specify a label for the key and, if prompted, select the key to import. Then click **Next**.
9. A summary panel is displayed. Click **Finish** to exit the wizard.
10. Repeat these steps to import all the keys and certificates you will use in the federation.

What to do next

Next you will want to provide your keys to your partner. See “Providing certificates to your partner.”

Providing certificates to your partner

Depending on the requirements of your environment, you might need to provide a key to your partner.

Before you begin

Use your “Planning message-level security” on page 32 to determine which certificates you might need to provide to your partner. In general, the public keys you will need to provide include:

Validation key
If you sign messages or assertions and your partner will validate those signatures, you must provide the public key that corresponds to the key that you used to sign.

Encryption key
For your partner to encrypt data to you, you must provide a public key to your partner. Your partner will use the public key to encrypt and you will use its corresponding private key to decrypt.

About this task

Usually you will provide your validation and encryption key in a metadata file that you will create and provide to your partner. In addition to the keys, other information about you, such as company name, and so on, is included in the metadata file. You will create this file later in the configuration process. For more information, see “Exporting certificates to a metadata file.”

In a SAML 1.0 federation, you also have the option of providing this information to your partner manually. See “Exporting a certificate.” You could also use the manual method if you have already provided your metadata to your partner and you need to provide an updated certificate by itself.

Exporting certificates to a metadata file

If you will be supplying your partner with a metadata file of your federation configuration, your public keys will be part of that file.

About this task

Depending on the message-level security and the SAML specification that you and your partner are using in the federation, the metadata file should include one or more of the following public keys:

- Key the partner will use for validating signed assertions, if you sign assertions
- Key the partner will use for validating signed messages, if you sign messages
- Key the partner will use for encrypting messages to you (in a SAML 2.0 federation)

Refer to “Planning message-level security” on page 32.

If your partner is using Tivoli Federated Identity Manager, you can export your configuration to a metadata file, including your keys, and your partner can import the file.

If you choose this way to provide your keys to your partner, you will want to export the metadata as part of establishing your federation. To continue, complete the remaining tasks in this chapter.

Exporting a certificate

About this task

Use this task to export your certificates if you cannot provide a metadata file containing your keys to your partner.

Procedure

1. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
2. Select the appropriate keystore from the Keystore table. You are prompted for your keystore password.
3. Type the password and click **OK**. The **View Keys** button is active
4. Click **View Keys**. The Keys panel is displayed. Keys in the selected keystore are listed.
5. Select the keys you want to export and click the **Export** button. The Export Key panel is displayed.

6. Select the format of the key you are exporting.

   (PEM)
   (Privacy-Enhanced Message) Public certificate

   PKCS#12
   Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard
7. Ensure that the **Include Private Key** check box is *not* selected. Only you should have your private key.
8. Click **Download Key**.
9. When prompted, enter a file name for the exported key.

   For example: `mypublickey.pem`

   Optionally, you can click **Browse** to find the file on the file system.
10. Click **Cancel** to exit.

Updating the cryptography policy

Use of encryption technology is controlled by United States law. IBM Java Solution Developer Kits (SDKs) include strong but limited jurisdiction policy files. To use PKCS#12 files with Tivoli Federated Identity Manager, you must first obtain the unlimited jurisdiction Java Cryptography Extension (JCE) policy files.

About this task

To review the security information for IBM Java SDKs, access the following URL: `http://www.ibm.com/developerworks/java/jdk/security/index.html`

To obtain the unlimited jurisdiction policy files:

Procedure

1. Update WebSphere with unrestricted Java Cryptography Extension (JCE) policy files. Access: http://www.ibm.com/developerworks/java/jdk/security/index.html
2. Select the link to the SDK that matches your environment, for example, for Java 1.5, the SDK is J2SE 5.0. You will see a page that displays the heading Security Information.
3. Select the link: **IBM SDK Policy Files**.

   Note: After you click this link, you will be redirected to the policy file in the SDK that is compatible with your version of Java; note, however, that the version number of the SDK might not be the same as the version number of the Java version you are using. For example, for Java 1.5 you might be directed to the SDK 1.4.
4. You will be prompted to log in using your IBM user ID and password. If you do not have an IBM user ID and password, you will need to register. Follow the registration link on the login page.
5. Log in.
6. When prompted, select the .zip file for the version of Java you are using. Then click **Continue** to begin the download.
7. Unpack the .zip file. The JAR files are:
   - local_policy.jar
   - US_export_policy.jar

8. Place the files in the following directory:

   ```
   your_Java_runtime_installation_dir/jre/lib/security
   ```

   For example, your Java runtime might have been installed as part of the embedded version of WebSphere Application Server. In this case, the directory might be

   ```
   /opt/IBM/FIM/ewas/java/jre/lib/security
   ```

Removing default keystores

Default keystores and certificates are included with Tivoli Federated Identity Manager. If you have created your own keystores, you might want to delete the default keystores. However, this task is optional.

Procedure

1. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
2. Select **DefaultKeyStore**, and then click **Delete.** A message asks you to confirm that you want to delete the specified keystore.
3. Click **OK** to delete the keystore.
4. Select **DefaultTrustedKeyStore**, and then click **Delete.** A message asks you to confirm that you want to delete the specified keystore.
5. Click **OK** to delete the keystore.

Enabling certificate revocation checking

You can use IbmPKIX trust manager to determine the validity of server certificates. If you enable this function, the trust manager checks the certificate that is presented by the SSL server when the SOAP client establishes an SSL connection. If it finds that the certificate has been revoked, the federation operation being attempted will fail.

About this task

The following procedures are required to enable certificate revocation checking:

- “Enabling WebSphere for certificate revocation checking.”
- “Enabling the IbmPKIX trust manager” on page 47.

Enabling WebSphere for certificate revocation checking

Before you can enable certificate revocation checking for Tivoli Federated Identity Manager, you must enable settings in WebSphere Application Server.

About this task

The procedure you should follow for enabling the settings depends on whether you are using the embedded version of WebSphere Application Server or an existing version of WebSphere Application Server. Choose the appropriate procedure for your installation:

Embedded WebSphere Application Server
“Enabling CRC on embedded WebSphere Application Server” on page 46,

Existing WebSphere Application Server
“Enabling CRC on existing WebSphere Application Server” on page 46.

Enabling CRC on embedded WebSphere Application Server

If you are using the embedded version of WebSphere Application Server, you must enable the settings that are required for certificate revocation checking (CRC) before you can configure certificate revocation checking in your Tivoli Federated Identity Manager environment.

Before you begin

Attention: Use this procedure only if you have installed Tivoli Federated Identity Manager using the embedded version of WebSphere Application Server.

About this task

To enable the appropriate settings, complete the following steps:

Procedure

1. Open a command prompt.
2. Start the WebSphere Application Server wsadmin tool. From your WebSphere profile, type the appropriate command for your operating system to start the tool:

   Windows

   `wsadmin.bat`

   AIX®, Linux, HP-UX, or Solaris

   `wsadmin.sh`

   Note: For more information about the options that can be specified when you run the wsadmin tool, refer to the http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.
3. At the command prompt, run the following commands and replace *server1* with the name of your server:

   ```
   set jvm [$AdminConfig getid
     /Server:server1/JavaProcessDef:/JavaVirtualMachine:/]
    $AdminConfig modify $jvm {{genericJvmArguments
     "-Dcom.ibm.jsse2.checkRevocation=true
      -Dcom.ibm.security.enableCRLDP=true"}}
    $AdminConfig save
   ```
4. Restart WebSphere Application Server.

What to do next

Continue with the steps in “Enabling the IbmPKIX trust manager” on page 47.

Enabling CRC on existing WebSphere Application Server

If you installed Tivoli Federated Identity Manager on an existing version of WebSphere Application Server, you must enable the IbmPKIX trust manager before you can configure certificate revocation checking in your Tivoli Federated Identity Manager environment.

Procedure

1. Log in to the console for your WebSphere Application Server.
2. Click **Servers** → **Application Servers**.
3. Select your server.
4. Click **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.

5. Under Generic JVM Arguments, add the following text:

   ```
   Dcom.ibm.jsse2.checkRevocation=true
   Dcom.ibm.security.enableCRLDP=true
   ```
6. Restart WebSphere Application Server.

What to do next

Continue with the steps in “Enabling the IbmPKIX trust manager.”

Enabling the IbmPKIX trust manager

To enable certificate revocation checking, you must first enable Tivoli Federated Identity Manager to use the IbmPKIX trust manager.

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Runtime Node Management**.
2. The Runtime Node Management panel is displayed. Click **Runtime Custom Properties**. The Runtime Custom Properties panel is displayed.
3. Click **Create**. A list item is added to the list of properties with the name of **new key** and a value of **new value**.
4. Click **Create** again. Another list item is added to the list of properties with the name of **new key** and a value of **new value**.
5. Select one of the placeholder properties.
6. Type `com.tivoli.am.fim.soap.client.jsse.provider` in the **Name** field. Do not insert the space character in this field.
7. Type `JSSE2` in the **Value** field.
8. Select the next placeholder property.
9. Type `com.tivoli.am.fim.soap.client.trust.provider` in the **Name** field.
10. Type `IbmPKIX` in the **Value** field.
11. Click **OK** to apply the changes that you have made and exit from the panel.

Chapter 7. Setting up transport security

To protect the message as it is communicated (transported) between the partners, SAML recommends using Secure Sockets Layer (SSL) with server authentication and in some cases with mutual authentication.

About this task

In a Tivoli Federated Identity Manager environment, you can ensure transport security by enabling SSL on the WebSphere Application Server where the runtime and management services component is installed. In addition, if you will be a client in an SSL communication in which mutual authentication is required using a client certificate, you will also want to configure your client certificate.

The general steps for enabling server and client authentication include the following tasks:

Procedure

1. "Enabling SSL on the WebSphere Application Server."

> **Note:** If you are a service provider in a SAML 1.x federation, you will always be the *client* in an SSL configuration. Therefore, you do not need to configure *server SSL*. Refer to the steps for configuring client certificates in "Configuring your client certificates" on page 58.

   Enabling SSL on a server includes the following subtasks:

   a. "Creating a certificate request" on page 50.
   b. "Receiving a signed certificate issued by a certificate authority" on page 51.
   c. "Associating a certificate with your SSL configuration" on page 52.
   d. Optionally, you might want to complete the steps in "Deleting the default certificate" on page 53.
   e. "Extracting a certificate to share with your partner" on page 53.

2. "Configuring client authentication requirements" on page 54. Your authentication requirement options are:
   - No authentication
   - Basic authentication, in which a username and password are requested
   - Client certificate authentication
3. If you will act as a client in the federation and your partner requires a client certificate, you will also need to complete the steps in "Configuring your client certificates" on page 58.

Enabling SSL on the WebSphere Application Server

To ensure that messages are secure when they are communicated between the federation partners, you will want to enable SSL on your WebSphere Application Server where the runtime and management services component is installed.

Before you begin

Note: If you are a service provider in a SAML 1.x federation, you will always be the client in an SSL configuration. Therefore, you do not need to configure SSL on your server. Refer to the steps for configuring client certificates in "Configuring your client certificates" on page 58.

Creating a certificate request

To ensure SSL communication, servers require a personal certificate (also referred to as a server certificate) that is signed by a certificate authority (CA). You must first create a personal certificate request to obtain a certificate that is signed by a CA.

Before you begin

The keystore, which will contain the certificate request and later the certificate, must already exist. You can use the default WebSphere Application Server keystore, NodeDefaultKeyStore, or you can create a new keystore. For instructions on creating a new keystore, refer to the WebSphere Application Server 6.1 information center http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

About this task

Complete the following tasks in the console. If you need additional details, refer to the WebSphere Information Center topic about creating a certificate authority request.

Procedure

1. Log in to the console.
2. Click **Security** → **SSL certificate and key management**.
3. Under **Related items** on the right, click **Key stores and certificates** and then click the name of the keystore where you will store the certificate, for example, **NodeDefaultKeyStore**.
4. Click **Personal certificate requests** under Additional Properties.
5. Click **New**.
6. In the **File for certificate request** field, type the full path where you want the certificate request to be stored and a file name. The file will have an .arm extension. For example: `c:\servercertreq.arm` (on a Windows server).
7. Type an alias name for the certificate in the **Key label** field. The alias is the name you give to identify the certificate request in the keystore.
8. Type a common name value. The common name is the name of the entity that the certificate represents. The common name is frequently the DNS host name where the server resides.
9. In the **Organization unit** field, type the organization unit portion of the distinguished name.
10. In the **Locality** field, type the locality portion of the distinguished name.
11. In the **State or Province** field, type the state portion of the distinguished name.
12. In the **Zip Code** field, type the zip code portion of the distinguished name.
13. In the **Country or region** list, select the two-letter country code portion of the distinguished name.

14. Click **Apply** and then click **Save**. The certificate request is created in the specified file location in the keystore. The request functions as a temporary placeholder for the signed certificate until you manually receive the certificate in the keystore.

    Attention: Keystore tools (such as iKeyman and keyTool) cannot receive signed certificates that are generated by certificate requests from WebSphere Application Server. Similarly, WebSphere Application Server cannot accept certificates that are generated by certificate requests from other keystore utilities.
15. Send the certificate request .arm file to a certificate authority for signing. Each certificate authority has its own preferred method of receiving requests. Use the method recommended by the certificate authority to whom you will make your request.
16. Make a backup copy of your keystore file before you receive the certificate that you have requested. Use the path information of your keystore as shown in the console to locate the file. Then copy it to a new location for safe-keeping.

What to do next

Complete the process of obtaining a signed certificate for your server by receiving the certificate from the CA as described in “Receiving a signed certificate issued by a certificate authority.”

Receiving a signed certificate issued by a certificate authority

When a certificate authority (CA) receives a certificate request, it issues a new certificate that functions as a temporary placeholder for a CA-issued certificate. A keystore receives the certificate from the CA and generates a CA-signed personal certificate that WebSphere Application Server can use for SSL security.

Before you begin

The certificate request must have been created and must be in a WebSphere keystore as described in “Creating a certificate request” on page 50. Also, the certificate must have been received from the CA and placed on your computer so that you can receive it into the keystore.

WebSphere Application Server can receive only those certificates that are generated by a WebSphere Application Server certificate request. It cannot receive certificates that were requested using other keystore tools, such as iKeyman or keyTool.

About this task

Complete the following tasks in the console. If you need additional details, refer to the WebSphere information center http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp topic about receiving a certificate issued by a certificate authority.

Procedure

1. Log in to the console.
2. Click **Security** → **SSL certificate and key management** → **Manage endpoint security configurations**.
3. Click the name of your node on the **Inbound** tree.
4. Click the **Manage certificates** button.

5. Click **Receive a certificate from a certificate authority**.
6. Type the full path and name of the certificate file that you received from the certificate authority.
7. Select the default data type from the list.
8. Click **Apply** and **Save**. The keystore contains a new personal certificate that is issued by a CA. The SSL configuration is ready to use the new CA-signed personal certificate.

What to do next

Next, you will need to associate the certificate with your SSL configuration. See "Associating a certificate with your SSL configuration."

Associating a certificate with your SSL configuration

After you have added a signed certificate to your keystore, you will need to associate your server SSL configuration settings with that certificate.

About this task

When you install WebSphere Application Server 6.1 and Tivoli Federation Identity Manager, two SSL configurations are created on the WebSphere Application Server:

- NodeDefaultSSLSettings
- FIMSOAPEndpointSSLSettings

NodeDefaultSSLSettings is the default SSL configuration setting that is defined by WebSphere Application Server. This configuration setting is for the SSL policy for your WebSphere server. The FIMSOAPEndpointSSLSettings configuration is added by Tivoli Federated Identity Manager to enable you to have a separate SSL policy that is specifically for the communication of SOAP messages with your federation partner.

After installation, both configurations use the `default` self-signed certificate in the NodeDefaultKeystore.

When you request and receive a signed personal certificate, the settings for both SSL configurations are set to `none`.

You must manually specify the personal certificate you want to use in each SSL configuration. You could use the same certificate in each configuration. If you want to use a different certificate, follow the instructions for "Creating a certificate request" on page 50 and "Receiving a signed certificate issued by a certificate authority" on page 51 to create and receive the additional signed certificate and repeat these instructions.

Procedure

1. Log in to the console.
2. Click **Security** → **SSL certificate and key management**.
3. Under **Related items** on the right, click **SSL configurations**.
4. Click the name of the SSL configuration you want to configure. For example, click **NodeDefaultSSLSettings**.
5. Ensure that the **Keystore name** is displaying the keystore where your certificate is stored.

6. Click the **Get certificate aliases** button to ensure that all certificate aliases in your keystore are displayed.
7. In the **Default server certificate alias** field, select your signed certificate.
8. Click **Apply** and then **Save** when prompted to save the configuration to the master configuration. The SSL configuration will now use the new certificate.

What to do next

Repeat these steps to associate the other SSL configuration with the appropriate certificate. Then, continue with the instructions for deleting the default certificate in “Deleting the default certificate” to prevent it from being used inadvertently.

Deleting the default certificate

After you have received your personal signed certificate, delete the default key so that it won't be used inadvertently.

About this task

Attention: Ensure that none of your SSL configurations use the default key before continuing with this procedure. Refer to the instructions in “Associating a certificate with your SSL configuration” on page 52.

Procedure

1. Click **Security** → **SSL certificate and key management**.
2. Under **Related items**, click **Key stores and certificates**.
3. Click **NodeDefaultKeyStore**.
4. Under Additional Properties, click **Personal certificates**.
5. Select the check box next to the **default** certificate.
6. Click the **Delete** button.
7. Click **Apply** and then **Save**.

What to do next

Continue with the instructions for “Extracting a certificate to share with your partner” so that you can provide it to your partner.

Extracting a certificate to share with your partner

After you have added a signed CA certificate to your server, you will need to export a copy of that CA certificate with its public key and provide it to your partner.

Before you begin

The keystore and the personal certificate must already exist.

Procedure

1. Click **Security** → **SSL certificate and key management** → **Manage endpoint security configurations**.
2. Select your node on the **Outbound** tree.
3. Click **Manage certificates**.
4. Select the CA signed certificate and click **Extract** in the upper-right corner.

5. Type the full path where you want to extract for the certificate. Include a name for the certificate file in the path. The signer certificate is written to this certificate file. For example, in Windows, you might specify: `c:\certificates\local_cert.arm`
6. Select the default data type from the list.
7. Click **Apply** and then **Save**. The signer portion of the personal certificate is stored in the .arm file that you specified.

What to do next

Now you are ready to provide the file to your partner so that your partner can add your certificate to its truststore.

Note: If your partner is using Tivoli Federated Identity Manager, the partner must import your certificate into its Tivoli Federated Identity Manager truststore.

To complete your SSL configuration, continue with the steps for "Configuring client authentication requirements."

Configuring client authentication requirements

As part of your options for securing messages, you can require your partner to authenticate itself to your point of contact server.

About this task

Note: In a SAML 1.x federation, only the identity provider acts as the server; therefore, only the identity provider partner must configure a client authentication setting.

First, you must decide if you will require client authentication.

- If you will not require client authentication, refer to "Configuring access with no authentication."
- If you require client authentication, you have two options:
  - Basic authentication. See "Configuring basic authentication access" on page 55.
  - Client certificate authentication. See "Configuring access with client certificate authentication" on page 56.

Configuring access with no authentication

If you will not require client authentication from your partner, configure the SOAP authentication settings appropriately.

About this task

By default after installation, the endpoint security settings are set to **Allow unauthenticated users access to SOAP endpoints**.

Note: These instructions apply to standalone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see "Configuring IHS for client authentication" on page 69.

To ensure that this setting is selected:

Procedure

1. Log in to the console. Click **Tivoli Federated Identity Manager** → **Domain Management** → **Point of Contact**.
2. Select the point of contact server that you are using in your environment.
3. Click the **Advanced** button. The SOAP Endpoint Security Settings panel is displayed.
4. Ensure that the SOAP Port is correct in your configuration and that **Allow unauthenticated users access to SOAP endpoints** is selected.
5. Click **OK**.
6. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

If you are configuring a SAML 2.0 federation, continue with the steps for configuring your client certificate, "Configuring your client certificates" on page 58. If you are configuring a SAML 1.x federation, the task is complete.

Configuring basic authentication access

If you will require basic authentication from your partner, you will need to create a user in your user registry that represents your service provider partner.

Before you begin

Before beginning this task:

- Decide whether you will allow access to the endpoint by authenticated users individually or by authenticated users who are part of specific groups.
- Ensure that you know the username and password that you will require your service provider to use.

About this task

To configure basic authentication, complete the following steps.

Procedure

1. In your user registry, create a user with a name that reflects your service provider partner. For example, create a user with a username of `soapclient`.

   Note: Refer to user creation instructions for the user registry you have configured for your environment.
2. Your next step depends on whether you will allow individual authenticated users or authenticated users who are part of specific groups.
   - If you require basic authentication from individual users, repeat step 1 for each service provider user you need to configure. Then, proceed to step 3 on page 56.
   - If you require basic authentication from users in specific groups, create a group for the users and add the user you created in step 1 to the group. For example, create a group with a name of `soapgroup` and then add user `soapclient` to the group.

     Note: Refer to group creation instructions for the user registry you have configured for your environment.

3. Configure the SOAP authentication settings in the Tivoli Federated Identity Manager console:

   Note: These instructions apply to standalone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see “Configuring IHS for client authentication” on page 69.

   a. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.
   b. Select the point of contact server that you are using in your environment.
   c. Click the **Advanced** button. The SOAP Endpoint Security Settings panel is displayed.
   d. Ensure that the SOAP Port is correct in your configuration and select the appropriate option for your configuration:
      - If you require individual users to authenticate, select **Allow authenticated users access to SOAP endpoints**.
      - If you require users in specific groups to authenticate, select **Allow users in the specified group access to SOAP endpoints** and specify the group name in the **Group Name** field.
   e. Select **Basic Authentication**.
   f. Click **OK**.
   g. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

If you are configuring a SAML 2.0 federation, continue with the steps for configuring your client certificate, “Configuring your client certificates” on page 58.

If you are configuring a SAML 1.x federation, you have completed the task.

Configuring access with client certificate authentication

If you will require client certificate authentication from your partner, you will need to:

Before you begin

1. Configure WebSphere Application Server to recognize the client certificate.
2. Create a user and possibly a group to represent the service provider partner.
3. Configure Tivoli Federated Identity Manager to require authentication.

Before beginning this task:

- Ensure you have the public key certificate for the client certificate that your partner will use to access your artifact resolution endpoint.
- Ensure you have the common name attribute of the certificate that your partner will use to access your endpoint. (For example, if the DN of the certificate is "/C=US/ST=TX/L=AUSTIN/O=SERVICEPROVIDER/CN=soapclient," then the CN is "soapclient.")
- Decide whether you will allow access to the endpoint by authenticated users individually or authenticated users who are part of specific groups.

About this task

To configure client certificate authentication, complete the following steps.

Procedure

1. Copy the public key certificate that your partner will present for authentication to your WebSphere Application Server.

   Note: In these instructions the partner's certificate is named `partnerca.pem` and the directory to which the certificate was copied is named `/tmp`.
2. Log in to the console.
3. Select **Security** → **SSL Certificate and Key Management**.
4. Select **Key stores and certificates**.
5. Select **NodeDefaultTrustStore**.
6. Select **Signer certificates**.
7. Select **Add**.
8. Complete the fields with the appropriate information for the certificate. For example:
   - Alias: `CACert`
   - File name: `/tmp/partnerca.pem`
   - Data type: `Base64-encoded`
9. Click **OK**.
10. WebSphere must be able to map the client certificate presented by your partner to a user identity in your user registry, using the common name attribute of the certificate. You can see the common name attribute by clicking on the certificate in the console and locating its **Issue to** field.
    a. In your user registry, create a user with a name that reflects your service provider partner. For example, create a user with a username of `soapclient`.

       Note: Refer to user creation instructions for the user registry you have configured for your environment.

    b. Your next step depends on whether you will allow individual authenticated users or authenticated users who are part of specific groups.
       - If you require client certificate authentication from individual users, repeat step 10a for each service provider user you need to configure. Then proceed to step 11.
       - If you require client certificate authentication from users in specific groups, create a group for the users and add the user you created in step 10a to the group. For example, create a group with a name of `soapgroup` and then add user `soapclient` to the group.

         Note: Refer to group creation instructions for the user registry you have configured for your environment.
         Then proceed to step 11.
11. Configure the SOAP authentication settings in the Tivoli Federated Identity Manager console:

    Note: These instructions apply to standalone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see “Configuring IHS for client authentication” on page 69.

    a. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.

    b. Select the point of contact server that you are using in your environment.

c. Click the **Advanced** button. The SOAP Endpoint Security Settings panel is displayed.

d. Ensure that the SOAP Port is correct in your configuration and select the appropriate option for your configuration:

   - If you require individual users to authenticate, select **Allow authenticated users access to SOAP endpoints**.
   - If you require users in specific groups to authenticate, select **Allow users in the specified group access to SOAP endpoints** and specify the group name in the **Group Name** field.

e. Select **Client Certificate Authentication**.

f. Click **OK**.

g. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

If you are configuring a SAML 2.0 federation, continue with the steps for configuring your client certificate, "Configuring your client certificates." If you are configuring a SAML 1.x federation, you have completed the task.

Configuring your client certificates

If your partner requires client certificate authentication, you will need to create and import the certificate that you must present to authenticate and also export the certificate to your partner.

Retrieving the server certificate from your partner

If your partner has server authentication configured, you will need the public key from that server certificate and you will store it in a truststore used by your Tivoli Federated Identity Manager key service.

Before you begin

Before continuing with this procedure, ensure that you have a truststore prepared for storing the certificate. Refer to "Preparing the keystores" on page 29.

Procedure

1. Log in to the console.
2. Click **Tivoli Federated Identity Manager** → **Key Service**.

   The Keystores panel is displayed.
3. Select the truststore where you want to store the certificate in the Keystore table. The **View Keys** button is activated.
4. Click **Retrieve Certificate from SSL Connection**. The Password panel is displayed.
5. Type your truststore password and click **OK**.
6. Complete the fields to specify the host name and port name from which you will retrieve the certificate. Optionally, click the **Show Signer Info** to view the certificate before retrieving.
7. Complete the **Alias** field with the name you want to use for the certificate. Then, click **OK**. The certificate is added to the truststore.

What to do next

If you will act as a client in an SSL connection with your partner and your partner requires you to authenticate using a client certificate, continue with "Obtaining your client certificate."

Obtaining your client certificate

If you will act as a client in an SSL connection with your partner and your partner requires you to authenticate using a client certificate, you will need to obtain and configure the certificate and then share that certificate with your partner.

Before you begin

Before continuing with this procedure, ensure that you have a keystore prepared for storing the certificate. Refer to "Preparing the keystores" on page 29.

Procedure

1. Request a public/private key pair certificate from a certificate authority (CA):
   a. Log in to the console.
   b. Click **Tivoli Federated Identity Manager** → **Key Service**.

      The Keystores panel is displayed.
   c. Select a keystore from the Keystore table. The **View Keys** button is activated.
   d. Click **View Keys**. The Password panel is displayed.
   e. Type your keystore password and click **OK**.
   f. Click **Certificate Request**. The Create a certificate request panel is displayed.
   g. Complete the fields. Then click **OK**. A public/private key pair is added to the keystore and a file with the encoded BASE64 data is created. The temporary self-signed certificate will need to be replaced with the signed certificate from the CA.

   Return to these instructions when your CA notifies you that your signed certificate is ready.
2. Receive the signed certificate from the CA:
   a. Log in to the console.
   b. Click **Tivoli Federated Identity Manager** → **Key Service**.

      The Keystores panel is displayed.
   c. Select the keystore where the CSR was generated in the Keystore table. The **View Keys** button is activated.
   d. Click **View Keys**. The Password panel is displayed.
   e. Type your keystore password and click **OK**.
   f. Click **Receive Certificate from CA**.
   g. Select the location of the certificate that you received from the CA. Then click **OK**. The temporary self-signed certificate in the keystore will be replaced with the received signed certificate.
3. Provide the public key for this certificate to your partner:
   a. Log in to the console.
   b. Click **Tivoli Federated Identity Manager** → **Key Service**.

      The Keystores panel is displayed.

c. Select the appropriate keystore from the Keystore table. You are prompted for your keystore password.

d. Type the password and click **OK**. The **View Keys** button is active

e. Click **View Keys**. The Keys panel is displayed. Keys in the selected keystore are listed.

f. Select the keys you want to export and click the **Export** button. The Export Key panel is displayed.

g. Select the format of the key you are exporting.

   (PEM)
   (Privacy-Enhanced Message) Public certificate

   PKCS#12
   Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard

h. Ensure that the **Include Private Key** check box is *not* selected. Only you should have your private key.

i. Click **Download Key**.

j. When prompted, enter a file name for the exported key.

   For example: `mypublickey.pem`

   Optionally, you can click **Browse** to find the file on the file system.

k. Click **Cancel** to exit.

What to do next

Provide the certificate to your partner. The partner must ensure that:

- It has, in its truststore, the CA certificate from the CA who issued your certificate.
- Its server can get to the CA's certificate revocation list.

Chapter 8. Selecting a point of contact server

Tivoli Federated Identity Manager is not directly involved in user authentication or the creation of an application session. Instead, Tivoli Federated Identity Manager relies on a *point of contact server*.

The point of contact server is a proxy or application server that interacts with a user, performs the authentication and manages sessions. In a typical deployment, the point of contact is located at the edge of a protected network in front of a firewall, such as in a DMZ.

The point of contact server provides endpoints, which are the locations to and from which messages are sent and received. Each endpoint has a URL, so that the endpoints can be accessed by external users as Web sites on the Internet. The point of contact receives access requests and provides the authentication service. It serves as the first component capable of evaluating the authentication credentials of the user that is requesting access to the protected network. It also manages the users' session lifecycle, from session creation, to session access, to session deletion (such as in response to session logout services).

The choice of type of point of contact server to use is determined by the security architecture and network topology requirements. Tivoli Federated Identity Manager supports four options for the point of contact server:

- IBM WebSphere Application Server
- Tivoli Access Manager WebSEAL
- WebSEAL No ACLD
- Generic point of contact server
- A custom point of contact server

WebSphere as point of contact server

If you will use IBM WebSphere Application Server, your configuration options depend on whether you will be the identity provider partner or the service provider partner.

Identity Provider options

When you use IBM WebSphere Application Server as the point of contact server and you are the identity provider in a federation, you have the following options for the type of authentication to use:

- Forms authentication using any supported user registry
- SPNEGO (Simple and Protected GSSAPI Negotiation Mechanism) using TAI (Trust Association Interceptor) authentication and using Microsoft Active Directory as the user registry

Service Provider options

When you use IBM WebSphere Application Server as the point of contact server and you are the service provider in a federation, single sign-on is enabled using Lightweight Third-Party Authentication (LTPA). You have the following options for hosting applications that will be used in the federation that is configured in Tivoli Federated Identity Manager:

- IBM WebSphere Application Server, either the same server on which Tivoli Federated Identity Manager is installed or on a separate server running either WebSphere Application Server version 5.1 or 6.x.
- Microsoft Internet Information Services server 6.0 with the Tivoli Federated Identity Manager Web Server plug-in installed
- IBM HTTP Server 6.1 with the Tivoli Federated Identity Manager Web Server plug-in installed
- Apache HTTP Server 2.0 or 2.2 with the Tivoli Federated Identity Manager Web Server plug-in installed

Each of these options has specific requirements. For more information about these requirements, see “WebSphere as point of contact for identity providers” on page 69 and “WebSphere point of contact server for a service provider” on page 85.

WebSEAL as point of contact server

To satisfy the functional requirements for a point of contact server, Tivoli Federated Identity Manager can leverage the extensive authentication and authorization capabilities of Tivoli Access Manager. In environments that use Tivoli Access Manager, a WebSEAL server typically acts as the point of contact.

WebSEAL is most commonly used as a reverse proxy that can control access to extensive protected resources, through the establishment and management of WebSEAL junctions. WebSEAL receives access requests, and serves as the first component capable of evaluating the authentication credentials of the user that is requesting access to the protected network. In addition, the point of contact must handle Web session management for user sessions.

The federation creation wizard requires specification of a URL for point of contact servers. The wizard presents a field in which to enter the URL that provides access to endpoints on the Point of Contact server. The URL must contain the following elements:

- The communications protocol. Either HTTPS or HTTP for communications between the point of contact server and the user. Use of HTTPS is recommended for optimal security.

  Note that this value must match how you configured your point of contact server (WebSEAL).

  For example:

  ```
  https://
  ```
- The domain address of the WebSEAL server:

  For example:

  ```
  idp.example.com
  ```
- When using WebSEAL, the next element is name of the WebSEAL junction that services requests for single sign-on services. This can be any value, but must match the name of a junction on the WebSEAL server.

  For example:

  ```
  /FIM
  ```
- The final element is the string `/sps`. The element of the URL is defined by Tivoli Federated Identity Manager to name a WebSphere context for single sign-on services. The value of this string is fixed and cannot be changed.

These parts are combined to form a URL. For example:

```
https://idp.example.com/FIM/sps
```

Later in the federation configuration, the URL is extended further when you select a choice of single sign-on protocol and assign more specific endpoints for profiles such as login and logout. This means that this URL becomes part of a number of longer URL paths (endpoints) that are managed as Tivoli Access Manager protected objects.

WebSEAL No ACLD as point of contact server

Tivoli Access Manager deployments often include both a policy server (pdmgrd) and an authorization server (acld). Tivoli Access Manager requires a deployed policy server, but does not require an active authorization server. Tivoli Federated Identity Manager also requires only a deployed policy server. The WebSEAL point of contact server does not depend on an authorization server for any authentication or authorization services.

By default, the Default IVCred Module Instance in the product contacts the Tivoli Access Manager authorization server (also known as pdacld) to issue a credential. A skeleton credential is then built from the user name. This credential includes the groups (and Universal User IDs) for that user as defined in the user registry for Tivoli Access Manager. However, when you select WebSEAL No ACLD as the point of contact, the product does not use the authorization server to build credentials.

To configure the " WebSEAL No ACLD" point of contact profile:

1. Log in to console.
2. Select **Tivoli Federation Identity Manager** → **Configure Trust Service** → **Module Instances**.
3. Select **Default IVCred Token** and click **Properties**.
4. Clear **Enable Access Manager (IVCred) credential issuing (requires PDJRTE to be configured)**.
5. Click **OK**.

Note: If switching the point of contact back to a WebSEAL server with an authorization server, you must select **Enable Access Manager (IVCred) credential issuing (requires PDJRTE to be configured)**.

Generic point of contact server

The generic point of contact is an additional point of contact implementation provided by Tivoli Federated Identity Manager. It is a HTTP-headers based solution that provides administrators with the ability to modify their point of contact environments (for example, Apache) to set and read headers. This allows integration with Tivoli Federated Identity Manager without writing a custom point of contact server. The generic point of contact works pretty much the same as the WebSEAL point of contact server. The main difference is that headers names are used for the user information.

There generic point of contact server is included in the point of contact profiles that ship with Tivoli Federated Identity Manager. The administrator must enable it by selecting it on the console and setting it as active. The administrator can use the console to modify the header names used by each callback.

Custom point of contact server

A custom point of contact server is made up of several customized callback modules that define sign in, sign out, local ID, and authentication. A custom point of contact server might be appropriate in your environment if you want to integrate an existing authentication or Web access management application with Tivoli Federated Identity Manager. For example, a custom point of contact server would be useful in the following scenarios:

- If you have an existing single sign-on cookie token that is used throughout your existing enterprise, you could implement a custom point of contact server that uses a SignIn callback that sets that custom single sign-on domain cookie that conforms to your existing single sign-on strategy.
- If you have an existing Web access management product that exposes a custom API for asserting a user identity to the environment or retrieving the current user for the request. You could implement a point of contact server that uses a local identity callback (to retrieve the user for the transaction) or implement a custom point of contact server that uses a SignIn callback to assert the user identity to the environment, or implement a point of contact server that uses both types of callbacks.

Developing a custom point of contact server requires programming experience with developing callback modules and knowledge of Tivoli Federated Identity Manager programming concepts. Refer to the developerWorks® links in the information center at http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp.

When you have completed the development work, you will need to integrate the solution with your Tivoli Federated Identity Manager environment. For more information, see the *IBM Federated Identity Manager Administration Guide*.

Chapter 9. Configuring WebSphere as point of contact server

Tivoli Federated Identity Manager can be installed with either an embedded WebSphere server or into an existing WebSphere environment. When you install the embedded server, and use WebSphere as a point of contact server, the installation automates much of the configuration. When you install into an existing WebSphere environment, and want to use WebSphere as a point of contact server, you must manually configure the WebSphere and IHS servers to fit your deployment.

When configured as a point of contact server, WebSphere provides authentication services. The authentication services are specific to the federation role (identity provider or service provider).

Note: For WebSphere Application Server Version 6.0.2, WebSphere as a point of contact is not supported by Tivoli Federated Identity Manager.

See:

- "Using IBM HTTP Server with WebSphere as point of contact"
- "WebSphere as point of contact for identity providers" on page 69
- "WebSphere point of contact server for a service provider" on page 85

Using IBM HTTP Server with WebSphere as point of contact

WebSphere Application Server Network Deployment (ND) can be deployed either standalone or as part of a WebSphere cluster. In both cases, a typical deployment environment includes an IBM HTTP Server (IHS) that is positioned between the WebSphere server and external connections, such as those that come through a firewall or demilitarized zone (DMZ).

Deployment of the IHS typically includes configuration of Secure Socket Layer (SSL) connections, to secure both external connections and internal connections to the WebSphere servers. Successful deployment of a Tivoli Federated Identity Manager environment that will use WebSphere as a point of contact server requires that SSL is enabled on the IHS server.

Enablement of SSL on IHS requires the generation of an SSL key database and key. You can use the ikeyman utility to generate the necessary keys. If you have not enabled SSL on the IHS server, you must complete this task before configuring Tivoli Federated Identity Manager.

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Working with key database
- Securing with SSL communications

Adding an SSL port for a SOAP backchannel

Tivoli Federated Identity Manager single sign-on federations support configuration of certificate authentication or basic authentication between federation partners. When the deployment environment includes IHS, you must configure a SOAP backchannel to support these authentication methods.

You must add a virtual host to the IHS configuration. The configuration settings are typically located in the standard IHS configuration file. For example, on Linux or UNIX:

```
/opt/IBM/HTTPServer/httpd.conf
```

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server:,. See the topics that describe how to secure an IBM HTTP Server, including:

- Securing with SSL communications

Updating federation configuration for SOAP connection

When the IBM HTTP Server is configured to listen on both the default port and the SOAP backchannel port, you need to define and configure your federations using those ports for the federation URLs.

Federation URLs should use port 443. This port is the default HTTPS port, so there is no need to include the actual port in the URL syntax. The SOAP backchannel port is typically 9444.

Since the SOAP backchannel security involves a connection with the IHS server, the typical configuration steps when defining a federation does not include specifying client authentication on the SOAP backchannel.

Note that the WebSphere environment can include the configuration of SSL between IHS and the nodes in the WebSphere cluster. See the WebSphere documentation if this configuration is appropriate for your deployment.

Confirming WebSphere Application Server security properties

If you installed the embedded version of WebSphere Application Server with the installation of the runtime and management services component, several of its settings were configured during installation. If you are using an existing version of WebSphere Application Server (such as a previously installed version or the separately installable version), you must configure these settings manually.

Before you begin

The settings are:

- Application and administration security are enabled.
- Single sign-on (LTPA Cookie) is enabled.

Use the following procedures to confirm that the configuration settings are correct for your Tivoli Federated Identity Manager environment.

Use the WebSphere management console to check the WebSphere settings.

About this task

Application and administration security is enabled

To confirm that application and administration security are enabled:

1. Click **Security** → **Secure administration, applications and infrastructure**.
2. Confirm that both administrative and application security are enabled.

Single sign-on is enabled

To confirm that single sign-on is enabled:

1. Click **Security** → **Secure administration, applications and infrastructure**.
2. Expand **Web security** on the right to display:
   - **General settings**
   - **single sign-on**
   - **Trust association**
3. Click **single sign-on**.
4. Ensure that **Enabled** is selected.
5. Navigate to **Security** → **Secure administration, applications and infrastructure > Web security - General settings**
6. On the Configuration tab, in the General Properties section, select the check box **Use available authentication data when an unprotected URI is accessed**.

Enabling multiple language encoding on WebSphere Application Server

Enable multiple language encoding by enabling UTF-8 client encoding in WebSphere Application Server.

About this task

The procedure for enabling multiple language encoding is the same for the embedded version of WebSphere Application Server and on an existing WebSphere Application Server.

Procedure

1. Open a command prompt.
2. Start the WebSphere Application Server wsadmin tool. From your WebSphere profile, type the appropriate command for your operating system to start the tool:

   Windows
   ```
   wsadmin.bat
   ```
   AIX, Linux, HP-UX, or Solaris
   ```
   wsadmin.sh
   ```
   Note: For more information about the options that can be specified when you run the wsadmin tool, refer to the http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.
3. At the command prompt, run the following commands to enable UTF-8 encoding:
   a. To show the current JVM properties:

```
$AdminTask showJVMProperties { -propertyName genericJvmArguments }
```

b. To set the JVM properties:

```
$AdminTask setGenericJVMArguments { -genericJvmArguments
"<current-jvm-properties>  -Dclient.encoding.override=UTF-8" }
```

c. To save the configuration changes:

```
$AdminConfig save
```

4. Restart WebSphere Application Server.

Mapping application roles to users

Before you begin

When IBM Tivoli Federated Identity Manager is deployed with embedded WebSphere, the IBM Tivoli Federated Identity Manager installation automatically maps application roles to users. WhenIBM Tivoli Federated Identity Manager is deployed with an existing WebSphere server, IBM Tivoli Federated Identity Manager, you must manually create the mappings.

You can specify the different roles depending on the security needs of your deployment.

About this task

Use the WebSphere administration console to specify the mappings

Procedure

1. Navigate to **Enterprise Applications > ITFIMRuntime > Security role to user/group mapping**
2. Select the mappings in the table of roles.

   For each role, select either Everyone or All authenticated.

   Note: FIMAnyAuthenticated must *not* be mapped to Everyone.

   Example roles:
   - TrustClientRole
   - FIMUnauthenticated
   - FIMSoapClient
   - FIMAnyAuthenticated
   - FIMAdministrator
   - TrustClientInternalRole
   - FIMNobody
3. Click **OK** when you are finished.
4. Synchronize all nodes in the cluster.

   For instructions, consult the WebSphere information center: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. See the topic *Mapping user to roles*.

Results

The Tivoli Federated Identity Manager runtime is now functional with WebSphere as a point of contact server in a WebSphere Network Deployment (ND) environment.

Configuring IHS for client authentication

When you configure partners for a single sign-on federation, you can specify the supported methods for client authentication. The federation partner GUI wizard will prompt you to specify either SSL certificate authentication or basic authentication. Based on your choice, you must configure IBM HTTP Server (IHS) appropriately. Complete the instructions in the following section for your authentication method.

Configuring certificate authentication for IHS

When the federation partner configuration includes SSL client certificate over a SOAP connection, you must import that certificate as Trusted certificate authority (CA) on the key database used by IHS for SSL.

For example, a key file database on Linux or UNIX is:

```
/usr/IBM/HTTPServer/conf/httpkeys.kdb
```

Use the **ikeyman** utility to import the certificate.

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Storing a certificate authority certificate

Configuring basic authentication for IHS

When the federation partner configuration includes basic authentication over a SOAP connection, you must enable LDAP authentication for IHS.

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Authenticating with LDAP on IBM HTTP Server

WebSphere as point of contact for identity providers

If you will be the identity provider in your federation and you are using IBM WebSphere Application Server as your point of contact server, you have two options for the authentication method you can use. Your choice of the authentication method will determine the requirements you will have in your environment.

Choose one of the following options for the authentication method on your WebSphere Application Server:

- Form-based authentication using any user registry that is supported by WebSphere Application Server
- Windows desktop authentication using the WebSphere Application Server 6.1 SPNEGO TAI support and Microsoft Active Directory as the user registry

Attention: Before proceeding with the tasks described in this chapter, confirm that your settings are correct using "Confirming WebSphere Application Server security properties" on page 66.

Form-based authentication

In this configuration, the identity provider uses any user registry that is supported by WebSphere Application Server with form-based authentication to authenticate users who are requesting single sign-on. All of the identity provider's users must exist in the supported user registry. When users try to use single sign-on to access a resource (such as a Web application), Tivoli Federated Identity Manager presents a login form. The login form is provided with Tivoli Federated Identity Manager.

An unauthenticated user who triggers a single sign-on request to a service provider resource will be authenticated against the configured WebSphere Application Server user registry.

An example of this configuration is shown in Figure 1.

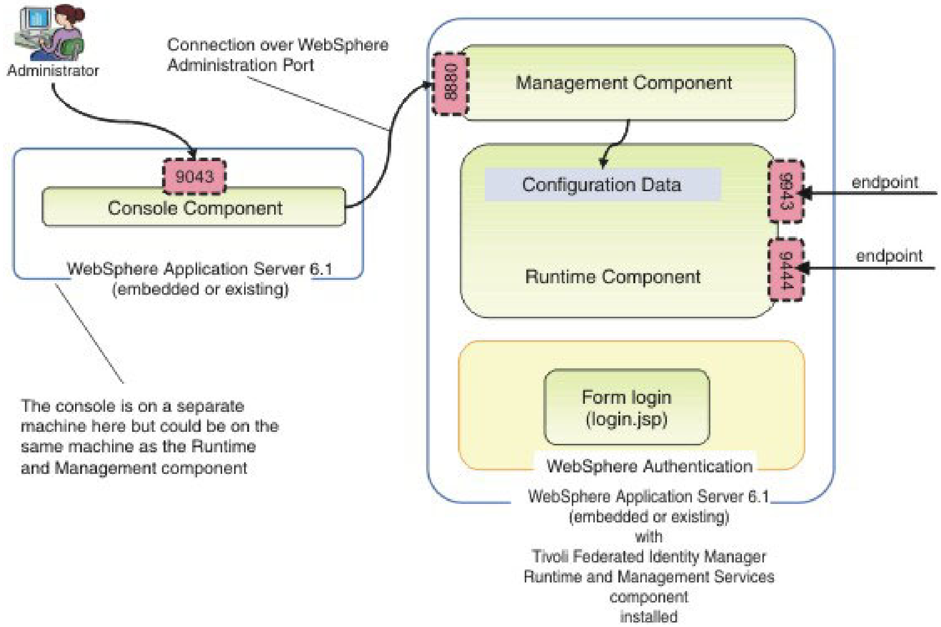

Administrator
Connection over WebSphere Administration Port
8880
Management Component
9043
Console Component
WebSphere Application Server 6.1 (embedded or existing)
Configuration Data
9943
endpoint
9444
endpoint
Runtime Component
The console is on a separate machine here but could be on the same machine as the Runtime and Management component
Form login (login.jsp)
WebSphere Authentication
WebSphere Application Server 6.1 (embedded or existing) with Tivoli Federated Identity Manager Runtime and Management Services component installed

Figure 1. Example of WebSphere Application Server with form-based authentication

Notes on configuration:

- Note that the WebSphere Application Server can be either an existing WebSphere deployment (with the correct level of fix pack applied) or can be the Embedded version of WebSphere Application Server Version 6.1 that is distributed with Tivoli Federated Identity Manager.
- A login form presented by the WebSphere Application Server where Tivoli Federated Identity Manager is installed. The login form is provided.

Complete the tasks in "Configuring form-based authentication" on page 72.

Windows desktop authentication through SPNEGO TAI with Microsoft Active Directory

This configuration uses a WebSphere Trust Association Interceptor (TAI) that supports a silent authentication using the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) protocol, which is provided with WebSphere Application Server. This configuration enables Tivoli Federated Identity Manager to securely acquire the user's desktop identity, which is then used to create the assertion for the federated single sign-on.

The identity provider uses Microsoft Active Directory as the user registry and Microsoft Windows Domain authentication. Windows must be configured as a domain controller. All of the identity provider's users must exist in the Active Directory user registry. To single sign-on to a Web application, the users use their Windows desktop credentials.

An example of this configuration is shown in Figure 2.

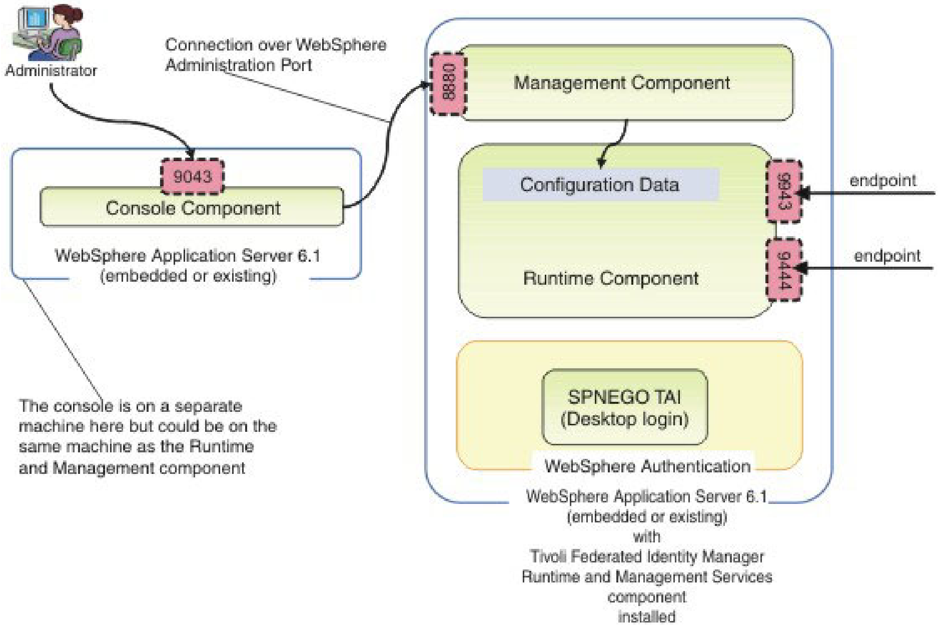

Administrator
Connection over WebSphere Administration Port
8880
Management Component
9043
Console Component
WebSphere Application Server 6.1 (embedded or existing)
Configuration Data
9943
endpoint
9444
endpoint
Runtime Component
The console is on a separate machine here but could be on the same machine as the Runtime and Management component
SPNEGO TAI (Desktop login)
WebSphere Authentication
WebSphere Application Server 6.1 (embedded or existing) with Tivoli Federated Identity Manager Runtime and Management Services component installed

Figure 2. Example of WebSphere Application Server with SPNEGO TAI authentication

Configuration notes:

- Note that the WebSphere Application Server can be either an existing WebSphere deployment (with the correct level of fix pack applied) or can be the Embedded version of WebSphere Application Server Version 6.1 that is distributed with Tivoli Federated Identity Manager.
- Microsoft Active Directory must be used as the user registry. Use a version that is supported by Microsoft Windows Server 2003. The user registry must include a user for the WebSphere administrative user and a user for the Kerberos identity. In addition, a keytab file must be built for each user. You will need the LDAP connection properties for the Active Directory server prior to configuring Tivoli Federated Identity Manager.

The user registry must be also be configured before configuring IBM WebSphere Application Server.

- SPNEGO authentication is provided with WebSphere Application Server in a Trust Association Interceptor (TAI) plug-in. It uses Kerberos to perform the authentication.
- The users log in using their desktop login to the Windows domain. This login method can also be referred to as "desktop single sign-on."
- The users' browsers must be configured so that Integrated Windows Authentication is enabled.

Complete the tasks in “Configuring SPNEGO authentication” on page 75.

Configuring form-based authentication

If you are using WebSphere Application Server as your point of contact server with form-based authentication, there are several configuration tasks that you will need to complete.

About this task

The tasks include:

1. “Selecting and installing the user registry”
2. “Configuring the user registry” on page 73
3. “Adding single sign-on users” on page 73
4. “Adding administrative users” on page 73
5. “Configuring user registry for embeddedWebSphere” on page 74
6. “Configuring an SSL connection to the user registry” on page 74
7. “Customizing the login form” on page 75

Selecting and installing the user registry

A user registry is required in your identity provider environment. The user registry is used as the repository for information about the users to whom you are providing single sign-on capabilities and the service providers with whom you have a federation. The user registry can also be used as the repository for information about the administrative users in your environment or you can choose to keep administrative users in a separate user registry.

Before you begin

You must choose a user registry that is compatible for use with your IBM WebSphere Application Server point of contact server and with the authentication method you will use.

If you are using form-based authentication, you can choose a user registry from many options. Refer to the WebSphere Application Server 6.1 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. Then locate information about selecting a user registry by selecting **WebSphere Application Server (Distributed platforms and Windows)** → **Securing applications and their environment** → **Authenticating users** → **Selecting a registry or repository**.

About this task

If you are using an existing installation of WebSphere Application Server, you might have a compatible user registry already installed and configured.

If you are using a new installation of the embedded version of WebSphere Application Server, you have the following options:

- Use the default file-based user repository realm (the federated repository), which was installed with the embedded version of WebSphere Application Server. The administrative user was configured in this registry during installation. Additional tasks needed for adding the single sign-on users are provided later in this chapter.
- Use a different user registry. Review the WebSphere Application Server documentation for information about your user registry options. Then, install and configure the user registry you chose, if you are not using a previously existing user registry. Then configure WebSphere to use that user registry. Refer to “Configuring user registry for embeddedWebSphere” on page 74.

Configuring the user registry

The configuration of your user registry is an important step in the overall configuration.

Before you begin

Before continuing with this task, you should have already selected which user registry you will use and have installed it as described in “Selecting and installing the user registry” on page 72.

About this task

In your user registry, you will create users to whom you are providing single sign-on capabilities. You can also create users for the administrators in your environment or you can choose to keep administrative users in a separate repository.

Adding single sign-on users:

In the identity provider environment, the user registry is used to authenticate the users who will use single sign-on. Add these users to your user registry using the documentation for your user registry.

Adding administrative users:

If you installed the embedded version of WebSphere Application Server, a file-based user repository realm, referred to as a *federated repository* was configured for the administrative users of Tivoli Federated Identity Manager. If you would prefer to manage administrative users through the same user registry where your single sign-on users are configured, you must add them to that user registry.

Before you begin

The administrative user that you specified during installation was created in the default user repository during the installation of Tivoli Federated Identity Manager.

About this task

To add this user to a different user registry:

Procedure

1. Create the user using the documentation for your user registry. Consider using the name ID and password that was used for the administrator when Tivoli Federated Identity Manager was installed.
2. Complete the instructions in "Configuring user registry for embeddedWebSphere."

Configuring user registry for embeddedWebSphere

If you installed the embedded version of WebSphere Application Server, the federated repository was configured as your user registry. If you want to use a user registry other than the default federated repository, you will need to modify the WebSphere Application Server settings.

About this task

To enable WebSphere to use your user registry:

Procedure

1. Log in to the console. Select **Security** → **Secure administration, applications, and infrastructure**. The Configuration tab is displayed.
2. Click on **Security Configuration Wizard** to change the user registry used by the WebSphere runtime.
3. The **Specify extent of protection panel** is displayed. Verify that the check box **Enable application security** is selected. Click **Next**.
4. The **Secure the application serving environment** panel is displayed. Select the appropriate option for the user registry you will use:
   - **Federated repositories**
   - **Standalone LDAP registry**
   - **Local operating system**
   - **Standalone custom registry**
5. Click **Next**. The **Configure user repository** panel is displayed. Specify values for each of the registry configuration settings. Refer to the online help for descriptions of the fields presented.
6. Click **Next** and finish the wizard. Save your configuration changes.
7. Stop and then restart the WebSphere Application Server. You must use the same administrative name you used to log in and make these changes.
8. From the console, select **Tivoli Federated Identity Manager** → **Manage Configuration** → **Domain properties**.
9. In the WebSphere Security section of the panel, update the following values:

   Administrative user name
   Replace the existing entry with the LDAP administrator account name that you entered in the previous step. For example, `ldapadmin`

   Administrative user password
   Enter the password for LDAP administrator.
10. Save the changes.
11. Stop the WebSphere Application Server.
12. Restart the WebSphere Application Server.

Configuring an SSL connection to the user registry

After you have configured your user registry, consider enabling SSL to protect the connection between it and the server.

About this task

For instructions, refer to the WebSphere Application Server 6.1 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. Locate information about creating SSL connections by selecting **WebSphere Application Server (Distributed platforms and Windows)** → **Securing applications and their environment** → **Securing communications**.

You might also need to refer to the documentation for your user registry.

Example

Customizing the login form

If you are using form-based authentication to authenticate the single sign-on users, a login form and an error page to the login form are provided for you to use.

About this task

The login form and error page are part of the response pages that are generated by Tivoli Federated Identity Manager. You can customize the pages to suit your environment needs and to modify their appearance. The page identifiers for these pages are:

proper/login/formlogin.html
The login page is displayed on the Web client side when single sign-on is initiated at the identity provider by an unauthenticated user.

proper/login/formloginerror.html
On authentication failure, the error page is displayed.

Configuring SPNEGO authentication

If you are using WebSphere Application Server as your point of contact server with SPNEGO authentication, there are several configuration tasks that you will need to complete.

About this task

These configuration tasks include:

Procedure

1. Configuring the Microsoft Active Directory, including:
   a. Creating an Active Directory user for the WebSphere administrative user.
   b. Creating an Active Directory user that will contain the Service Principal Name (SPN) of the Tivoli Federated Identity Manager server.
   c. Building a Kerberos keytab file and assigning the SPN for the Active Directory user created in 1b.
   d. Collecting the Active Directory configuration parameters.
2. Configuring the Windows domain and user logins.
3. Configuring WebSphere Application Server, including:
   a. Configuring administration security, with Active Directory used as the type of LDAP user registry.
   b. Configuring an SSL connection to Active Directory (optional).

4. Enabling WebSphere SPNEGO and the Trust Association Interceptor (TAI), using the Integrated Solutions Console. Optionally, you can customize the TAI attributes, as might be required in your environment.
5. Instructing your users to configure Internet Explorer, as follows:
   a. Adding the hostname as a trusted host in the Intranet Zone.
   b. Enabling Integrated Windows Authentication.

Configuring Active Directory for use with SPNEGO

If you will use WebSphere Application Server with SPNEGO authentication, you must use Microsoft Active Directory as your user registry. You will need to perform several configuration tasks in Microsoft Active Directory:

Before you begin

- Create a user for the WebSphere administrative user.
- Create a user that will contain the Service Principal Name (SPN) of the Tivoli Federated Identity Manager server.
- Build a Kerberos keytab file and assign the SPN to the Active Directory user that was created for that purpose.
- Collect Active Directory connection parameters.

Microsoft Active Directory is a required component in an identity provider environment in which IBM WebSphere Application Server with SPNEGO authentication is used as the point of contact server. Your Microsoft Active Directory should be installed and configured for your network before you begin this task.

About this task

For the details of completing the steps in this procedure, you will need to refer to the Microsoft Active Directory documentation.

Procedure

1. Using the Active Directory Users and Computers Console, create an Active Directory user for the WebSphere administrative user. This user is a regular user account in Active Directory, with no special account privileges. Use a user name that reflects the role of this user. For example, consider using `wasadmin`.
2. Using the Active Directory Users and Computers Console, create a user that will contain the Service Principal Name (SPN) of your Tivoli Federated Identity Manager server. The user name for this account is not important. This user's Service Prinicpal Name will be set using the ktpass utility in a subsequent step. Give this user a very secure password and set the password to never expire.
3. Use the ktpass command to build a keytab file for the WebSphere Kerberos user. The ktpass utility is included with the Microsoft Windows 2003 Server Support Tools package. Use the following parameters with the command:

Table 8. Parameters to use with the Microsoft Windows ktpass command

Parameter	Example value	Description
-out	`was1-krb5.keytab`	A filename in which to store the secret key that will later be used for Kerberos authentication validation on the WebSphere server. This file will be uploaded to the WebSphere server when you enable SPNEGO. See "Enabling and configuring SPNEGO authentication" on page 81.
-princ	`HTTP/ibm-fim611-1.fimtest. example.com@FIMTEST .EXAMPLE.COM`	The Kerberos service principal name to use for generating the key. This is case sensitive and MUST start with `HTTP/`. The portion following the `HTTP/` must be the fully qualified DNS domain name of the URL that users will see on their browsers when accessing the WebSphere server.
-pass	`*`	The password to set for the Kerberos principal. A value of `*` will result in prompting for the password. The password must match the user created in step2 on page 76.
-mapuser	`was-1`	The Active Directory user to whom the Kerberos service principal will be mapped. The value here should match the user name you created in step2 on page 76.
-mapOp	`set`	Indicates that the SPN should overwrite any existing value mapped for this Active Directory user.

The following example shows an execution of the ktpass command. It also shows the use of the setspn command to list service principal names for the was-1 user, for information and verification purposes.

```
C:\Program Files\Support Tools>ktpass -out was1-krb5.keytab
 -princ HTTP/ibm-fim611-1.fimtest.example.com@FIMTEST.EXAMPLE.COM
 -pass * -mapuser was-1 -mapOp set

Targeting domain controller: ibm-fimtest-ad.fimtest.example.com

Successfully mapped HTTP/ibm-fim611-1.fimtest.example.com:

Type the password again to confirm:

Key created.

Output keytab to was1-krb5.keytab:

Keytab version:0x502

keysize 76 HTTP/ibm-fim-611-1.fimtest.example.com@FIMTEST.EXAMPLE.COM
 ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x3 (DES-CBC-MD5)
 keylength 8 (0x799b26bfe9ad3ba4)

Account was-1 has been set for DES-only encryption.

C:\Program Files\Support Tools>setspn -1 was-1

Registered ServicePrincipalNames for
CN=was-1,CN-Users,DC=fimtest,DC=ibm,DC=com:

   HTTP/ibm-fim611-1.fimtest.ibm.com
```

Figure 3. Example of the ktpass command

The keytab file that is created in this step will be uploaded to the Tivoli Federated Identity Manager server during the configuration of WebSphere Application Server, see “Configuring WebSphere for use with SPNEGO” on page 79.

4. Collect Active Directory connection configuration information to use in the WebSphere Application Server configuration, as follows:
   a. Locate the following information in the Active Directory LDAP tree:

      Hostname
      The hostname of the Active Directory server.

      Port Port number of the active directory server.

      Base DN
      The base search DN for active directory users.

      Bind DN
      An administrative user's active directory DN for performing LDAP searches. This value does not need to be the DN for the domain administration account but rather the DN for any valid active directory user.

      Bind password
      The password for the user represented by the Bind DN.

   b. If an SSL connection is required to Active Directory, WebSphere must be configured with the certificate of the domain controller's issuing CA. If Windows Certificate Services was installed on the domain controller, this will be the CA certificate of the Certificate Services on that domain controller. To export the CA certificate to a file:

1) Open **Administrative Tools** → **Certification Authority**. Then right-click on the top-level CA name, and click **Properties**.
2) Click the **General** tab and then click **View Certificate**.
3) Click the **Details** tab and click **Copy to File**.

   The file will be saved in DER encoded binary format. You will use this file as part of the WebSphere configuration, if SSL server authentication is needed to contact the Active Directory server through the LDAP/SSL interface.

Configuring the Windows domain and user logins

To use Windows desktop single sign-on, the users' desktop logins must be authenticated to the Windows domain.

About this task

Use of the Windows desktop single sign-on to the Tivoli Federated Identity Manager server requires that users log in to their desktop as members of a Windows domain. In particular, the Windows domain must support Kerberos authentication to a Microsoft Active Directory. Refer to Microsoft documentation for the details of creating this environment.

This configuration will enable the identity provider to support internal users who are connected to the identity provider's intranet using a desktop login made to a Windows domain. However, an identity provider might also want to support external users who do not have a Windows domain login. These external users would need to authenticate using a login form.

By default, the SPNEGO TAI support in Tivoli Federated Identity Manager displays a login form when a user who has not authenticated through the desktop login attempts a single sign-on. By default, the login form is the sample login form that is provided with Tivoli Federated Identity Manager. You can customize the appearance of this form, as described in "Customizing the login form" on page 75. If you do not want to display this login form, you can modify the TAI attributes as described in "Configuring custom TAI attributes" on page 84.

Configuring WebSphere for use with SPNEGO

Before you can use WebSphere Application Server with SPNEGO, you must configure WebSphere application security with Active Directory set as the user repository.

About this task

The steps include:

- (Optionally) Loading the CA root certificate of the Active Directory server to enable SSL between the server and Active Directory.
- Enabling WebSphere application security with Active Directory as the user registry.
- Configuring details for the standalone LDAP directory to point to the Active Directory server.

Procedure

1. Optional: Load the CA root certificate of the Active Directory server. This step is required only if you will use LDAP/SSL to communicate with the Active Directory server. Before continuing with this step, make sure you have

completed the steps in “Configuring Active Directory for use with SPNEGO” on page 76, especially step 4b on page 78. If you are not using LDAP/SSL, continue with the next step.

a. Log in to the console.

b. Click **SSL certificate and key management**.

c. On the SSL Certificate and key management panel, click **Key stores and certificates**.

d. On the Key stores and certificates panel, click **NodeDefaultTrustStore**.

e. On the NodeDefaultTrustStore panel, click **Signer certificates**.

f. On the Signer certificates panel, click **Add** to add a new signer.

g. Complete the details for the signer certificate and click **OK**. Use the following values:

Table 9. Signer certificate details in SPNEGO environment

Field name	Value
Alias	Any alias name for the CA certificate from Active Directory. For example, you might use the name of the Active Directory domain controller.
File name	The path and file name of the certificate. Note that this path and file name is on the server where WebSphere Application Server is installed, not on the server where the browser is running. This means that the file must be copied to the WebSphere server` prior to completing this step.
Data type	The file format for the certificate. Use the same format you used in step 4b on page 78.

After the certificate is successfully loaded, you should see it listed in the signer certificates list.

2. Enable WebSphere application security with Active Directory set as the user registry.

Note: To complete this step, you must be able to contact the Active Directory server using port 389 (that is, without using SSL). During this step, the security configuration wizard performs a connection test, which does not support SSL and the setup cannot proceed if this test fails. You can enable LDAP/SSL after the test and setup have been completed.

a. In the console, click **Security** → **Secure administration, applications, and infrastructure**.

b. Then click the **Security Configuration Wizard** button to start the Security Configuration wizard.

c. Click **Next**.

d. In step 1 of the wizard, make sure that the **Enable application security** check box is selected, and then click **Next**.

e. In step 2 of the wizard, select **Standalone LDAP directory** and then click **Next**.

f. In step 3 of the wizard, enter the following parameters and then click **Next**.

Table 10. Parameters for the LDAP directory in SPNEGO environment

Field name	Value
Primary administrative user name	Use the WebSphere administrative user name that was created in the Active Directory.

Table 10. Parameters for the LDAP directory in SPNEGO environment (continued)

Field name	Value
Type of LDAP server	Microsoft Active Directory
Host	The hostname of the Active Directory server. For example: `ibm-fimtest-ad.fimtest.example.com`
Port	Until you run the test step of this configuration wizard, use a port that does not use SSL. For example, use 389.
Base distinguished name (DN)	The base search DN for user entries. For example: `cn=users,dc=fimtest,dc=ibm,dc=com.`
Bind distinguished name (DN)	The DN of a valid Active Directory user. For example: `cn=administrator,cn=users,dc=fimtest,dc=ibm,dc=com.`
Bind password	The Active Directory password for the user represented by the bind DN.

g. In step 4 of the wizard, the connection to the Active Directory server is tested. Click **Finish** to complete the wizard.

3. Configure details for the standalone LDAP directory to point to the Active Directory server.
   a. In the console, click **Security** → **Secure administration, applications, and infrastructure**.
   b. In the Available realm definition list, select **Standalone LDAP registry** and then click **Configure**.
   c. Complete the details about your configuration, including SSL and SSL port if necessary, and then click **OK** and save your changes.

Enabling and configuring SPNEGO authentication

Before you can use WebSphere Application Server with SPNEGO, you must enable SPNEGO authentication in Tivoli Federated Identity Manager and configure its properties.

About this task

You will use the console to complete this procedure. This procedure includes:

- Enabling SPNEGO for use with Tivoli Federated Identity Manager.
- Configuring the WebSphere Kerberos client.
- Configuring the TAI properties file.
- Setting the JVM startup parameters.

Procedure

1. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.
2. Select the point of contact server profile that you are using in your environment.
3. Click the **Advanced** button. The SOAP Endpoint Security Settings panel is displayed.
4. Click **SPNEGO Authentication Settings**.
5. Select the **Enable SPNEGO Authentication** check box.
6. Complete the fields with the information for your authentication configuration. Refer to the online help for complete descriptions of the fields.

7. Import the Kerberos keytab file, which you created using the -out option of the ktpass utility, as follows:
   a. Click the **Import Keytab file** button.
   b. In the **Location of Keytab File** field, type the path for the file or optionally, use the **Browse** button to locate the file.
   c. Click **Finish**.
8. Click **OK**.

Configuring the Trust Association Interceptor

If you enable and configure SPNEGO using the console, TAI is automatically enabled in the WebSphere Application Server settings.

About this task

In general, no further configuration is necessary. The `tai.properties.template` file contains default values for all of the WebSphere SPNEGO TAI. For additional information about these values, refer to "SPNEGO TAI configuration attributes."

Note: If you plan to make changes to these default values, use the instructions in "Configuring custom TAI attributes" on page 84.

SPNEGO TAI configuration attributes:

The Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) custom configuration attributes control different operational aspects of the SPNEGO TAI. These attributes are stored in the `tai.properties.template` file.

Content

The file is located in the following default directory:

AIX, Linux or Solaris
`/opt/IBM/FIM/etc/tai.properties.template`

Windows
`C:\Program Files\IBM\FIM\etc\tai.properties.template`

In general, you should not need to modify this file. You can configure the TAI by using the console as described in "Configuring the Trust Association Interceptor." However, if you need to make additional changes that require updates to the `tai.properties.template`, use the instructions in "Configuring custom TAI attributes" on page 84.

Note: The version of the tai.properties.template file that is installed as part of Tivoli Federated Identity Manager contains additional attributes that are not provided with WebSphere Application Server 6.1. If your environment requires the use of attributes that are not described here, refer to the WebSphere Application Server 6.1 information center for a list of all the attributes available for the customization of SPNEGO TAI configuration. The information center is located at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.

The following figure describes the content of the `tai.properties.template`.

```
##########################################################
# Template properties files for SPNEGO TAI
#
# Where possible defaults have been provided.
#
##########################################################
#---------------------------------------------------------
# Hostname
#---------------------------------------------------------
com.ibm.ws.security.spnego.SPN1.hostName=@POCHOST@

#---------------------------------------------------------
# (Optional) SpnegoNotSupportedPage
#---------------------------------------------------------
com.ibm.ws.security.spnego.SPN1.spnegoNotSupportedPage=file:///@SPNEGOFAILED@

#---------------------------------------------------------
# (Optional) NTLMTokenReceivedPage
#---------------------------------------------------------
com.ibm.ws.security.spnego.SPN1.NTLMTokenReceivedPage=file:///@SPNEGOFAILED@

#---------------------------------------------------------
# (Optional) FilterClass
#---------------------------------------------------------
#com.ibm.ws.security.spnego.SPN1.filterClass=com.ibm.ws.spnego.HTTPHeaderFilter

#---------------------------------------------------------
# (Optional) Filter
#---------------------------------------------------------
com.ibm.ws.security.spnego.SPN1.filter=request-url%=/sps/wasauth

#---------------------------------------------------------
# (Optional) Credential Delegation
#---------------------------------------------------------
#com.ibm.ws.security.spnego.SPN1.enableCredDelegate

#---------------------------------------------------------
# (Optional) Credential Delegation
#---------------------------------------------------------
#com.ibm.ws.security.spnego.SPN1.trimUserName=
```

Figure 4. tai.properties.template file

Macros

The following macros are used in the `tai.properties.template` file.

Table 11. Macros used in the tai.properties.template file

Macro	Description	Default value
`@POCHOST@`	The fully qualified point of contact hostname. This hostname is used in the point of contact server URL.	`poc.example.com`
`@SPNEGOFAILED@`	The full path to an HTML file that is sent to the browser when SPNEGO authentication negotiation is unsuccessful. This HTML file automatically redirects the browser to the sample login page, that is provided with Tivoli Federated Identity Manager.	*installation_directory*`/etc/spnego_failed.html` This parameter cannot be configured using the console. The correct path is configured when SPNEGO is configured.

Configuring custom TAI attributes:
Before you begin

The TAI is enabled automatically when you enable SPNEGO using the console as described in “Enabling and configuring SPNEGO authentication” on page 81. However, if you need to customize TAI attributes, you will need to modify the tai.properties.template file.

About this task

Review the content of the tai.properties.template file in “SPNEGO TAI configuration attributes” on page 82.

Procedure

1. Locate the file and make a backup copy of it. The file is located in the following default directory:

   AIX, Linux or Solaris
   ```
   /opt/IBM/FIM/etc/tai.properties.template
   ```

   Windows
   ```
   C:\Program Files\IBM\FIM\etc\tai.properties.template
   ```
2. Open the file in a text editor.
3. Make the changes that are appropriate for your environment.
4. Save and close the file.

Configuring browsers for use with SPNEGO

Users must use desktop single sign-on to access the Tivoli Federated Identity Manager server after SPNEGO authentication is configured.

Before you begin

This requires that:

- The user's browser recognizes the Tivoli Federated Identity Manager server as an *intranet site*.
- The user's browser is enabled for Integrated Windows Authentication.

The instructions in this procedure are for Internet Explorer 6 and later. For other browser types, such as Mozilla, refer to the documentation for the browser.

About this task

In general, browser configuration for SPNEGO involves:

- Adding the hostname of the WebSphere Application Server that is used with Tivoli Federated Identity Manager to the local intranet list.
- Verifying that Integrated Windows Authentication is checked in the Advanced security settings of the browser.

Procedure

1. Add the hostname:
   a. Start Internet Explorer and click **Tools** → **Internet Options**.
   b. Click the **Security** tab and then click **Local intranet**.
   c. Click the **Sites** button. Make sure the **Include all local (intranet) sites not listed in other zones** is checked. Then click the **Advanced** button.

   d. Add the Web sites for the WebSphere Application Server as viewed at the browser, using either http or https, as needed.

      Note: This hostname must match the principal name configured for the keytab file.
      For example:

      ```
      http://ibm-fim611-1.fimtest.example.com
      https://ibm-fim611-1.fimtest.example.com
      ```

2. Verify that Integrated Windows Authentication is enabled:
   a. Start Internet Explorer and click **Tools** → **Internet Options**.
   b. Click the **Advanced** tab and scroll to the Security section.
   c. Ensure that the **Enable Integrated Windows Authentication (requires restart)** box is selected.
   d. Save the changes and restart the browser, if necessary.

WebSphere point of contact server for a service provider

If you will be the service provider in your federation, you have several options for your configuration.

If you use WebSphere Application Server as your point of contact server, you can use any of the following types of servers to host the target Web applications that your single sign-on users will access:

- IBM WebSphere Application Server 5.1 or 6.0 or later. (In most cases, you will host your Web applications on an installation of WebSphere Application Server that is separate from the server where Tivoli Federated Identity Manager is installed. However, if your installation of WebSphere Application Server is version 6.1 and meets the requirements for the installation of Tivoli Federated Identity Manager and for hosting your Web applications, you can use the same server for both Tivoli Federated Identity Manager and your applications.)
- Microsoft Internet Information Service 6.0
- IBM HTTP Server 6.1
- Apache HTTP Server 2.0 and 2.2

If you choose a server other than WebSphere Application Server as the host for your applications, you must install the Tivoli Federated Identity Manager Web server plug-in on your application server. The figure that follows shows an example of a Tivoli Federated Identity Manager environment in which applications are hosted by a separate Web server.

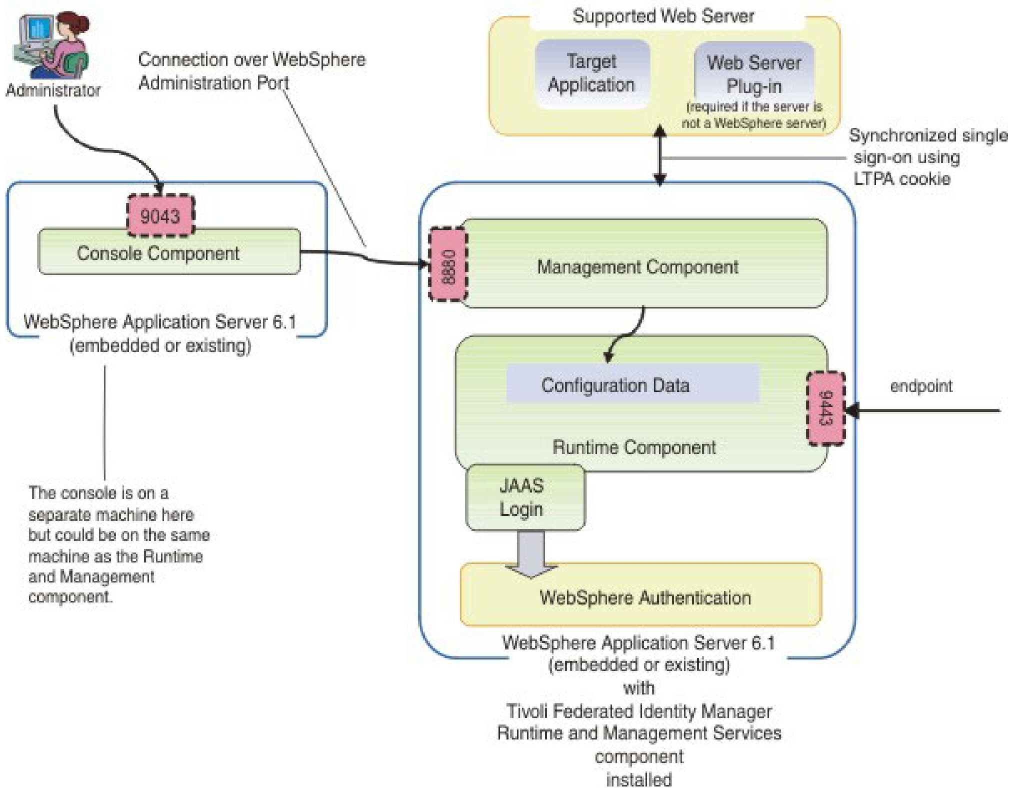

Administrator
Connection over WebSphere Administration Port
Supported Web Server
Target Application
Web Server Plug-in
(required if the server is not a WebSphere server)
Synchronized single sign-on using LTPA cookie
9043
Console Component
WebSphere Application Server 6.1 (embedded or existing)
8880
Management Component
Configuration Data
9443
endpoint
Runtime Component
JAAS Login
WebSphere Authentication
WebSphere Application Server 6.1 (embedded or existing) with Tivoli Federated Identity Manager Runtime and Management Services component installed
The console is on a separate machine here but could be on the same machine as the Runtime and Management component.

Figure 5. Example of Tivoli Federated Identity Manager and a Web application server

In the configuration depicted, the target application is hosted by a server that is separate from the Tivoli Federated Identity Manager server. The user authenticates to the identity provider and the credential is then transferred from the identity provider to Tivoli Federated Identity Manager, where the service provider validates the token and returns an LTPA cookie with the user's identity and any attributes carried by the token or added by the service provider's mapping rules. The user is redirected (by some single sign-on protocol) to the target application where the LTPA cookie is transferred from the Tivoli Federated Identity Manager node to the Web server node. The LTPA key must be shared between these nodes for the cookie to be recognized.

If the Web server is not a WebSphere Application Server, the Tivoli Federated Identity Manager Web server plug-in must be installed on that server. The plug-in extracts the identity and attributes from the LTPA cookie and provides it to the target application using one or more HTTP headers or server variables.

Environment requirements

Target applications can be hosted by any of the following servers:

- WebSphere Application Server 5.1 or 6.0 or later
- Microsoft Internet Information Server 6.0
- IBM HTTP Server 6.1
- Apache HTTP Server 2.0 and 2.2

Attention: If you host target applications on a server other than WebSphere Application Server, you must install the Tivoli Federated Identity Manager Web Server plug-in on that server.

- Applications must be able to accept user identity by way of an HTTP header or server variable.
- A user registry is required in your environment for both your point of contact server and your application's server. The users to whom you are providing single sign-on capabilities must exist in both user registries. If a separate server will host your target application, such as another WebSphere Application Server or a supported server with a plug-in such as an IHS, IIS, or Apache server, you will need to configure a user registry for that server also. Consider selecting a user registry that can be used for your point of contact server and your Web server to minimize the number of user registries you must maintain in your environment

Plug-in requirements

You must also ensure that your environment meets the following requirements:

- Applications must be able to accept user identity by way of an HTTP header or server variable.
- The user name for each single sign-on user must exist in both the WebSphere Application Server user registry (where Tivoli Federated Identity Manager is installed) and the user registry of the Web server.
- The Tivoli Federated Identity Manager server and the Web server must be in the same DNS domain and the LTPA cookie must be configured as a domain cookie.
- The LTPA key file and password must reside on both the Tivoli Federated Identity Manager server and the Web server where the plug-in is installed.

Configuring WebSphere

To configure your WebSphere Application Server point of contact server, continue with the instructions in “Configuring a WebSphere Application Server point of contact server (service provider).”

Configuring a WebSphere Application Server point of contact server (service provider)

If you are using WebSphere Application Server for your point of contact server, there are several configuration tasks that you will need to complete.

About this task

Attention: Before proceeding with the tasks described in this section, confirm that your settings are correct using “Confirming WebSphere Application Server security properties” on page 66.

Configuring the LTPA cookie

In general, federated single sign-on is available only if the applications share a common domain name with the service provider's assertion consumer service endpoint. To ensure that your applications share the proper domain name, you must configure the LTPA as a domain cookie, using the domain of your assertion consumer service endpoint.

Procedure

1. Log in to the console.
2. Click **Security** → **Secure Administration, Applications, and Infrastructure** → **Web Security**.

3. Click **single sign-on**.
4. To restrict the LTPA cookie to SSL sessions, select **Requires SSL**.
5. Specify the domain name in the Domain name field. Precede the domain name with a dot (.). Setting the domain name ensures that the LTPA cookie is made available to all of the Web servers in that specified domain.
6. Clear the **Interoperability Mode** check box. Interoperability mode results in two cookies (a version 1 LTPA cookie and a version 2 LTPA cookie) being placed on the browser. The Tivoli Federated Identity Manager Web Plug-in supports only version 2 LTPA cookies.
7. Click **OK** and then click **Save**.

What to do next

Additional information about setting the domain properly can be found in the topic about implementing single sign-on to minimize Web user authentications of the WebSphere Application Server 6.1 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.

Defining attributes for the LTPA token

By default, all available attributes will be included in the LTPA token. If you want to limit the attributes to specific ones, you must modify the attribute filtering settings on your WebSphere Application Server.

About this task

Note: Your target application must be configured to make use of the attributes that are included in the LTPA token. For more information about development topics, refer to the developerWorks links on the Welcome page of the information center at http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp.

Procedure

1. Log in to the console.
2. Click **Security** → **Secure Administration, Applications, and Infrastructure**.
3. Expand the list for **Java Authentication and Authorization Service**. Then, click **System logins**.
4. In the JAAS System logins panel, select **FIM_OUTBOUND**.
5. In the Additional Properties section of the FIM_OUTBOUND panel , select **JAAS login modules**.
6. Select the class name for the WebSphere point of contact attribute map login module.

   ```
   com.tivoli.am.fim.fedmgr2.was.jaas.login.
             WASPocAttributesMapLoginModule
   ```

   A list of configuration properties is displayed.

   Note: If you want to remove all attributes, click the check box next to **ssoAttributeNames** and click **Delete**. Otherwise, to modify the attributes, continue with the remaining steps.
7. Click **ssoAttributeNames** to see the default properties. The **ssoAttributeNames** setting is configured by default with an * in the **Value** field to specify that all attributes should be included in the token.
8. If you want to change the attributes, remove the * and type an attribute name, such as `AuthenticationMethod` or multiple attribute names, such as

`AuthenticationMethod,AuthenticationInstant`. If you specify multiple attributes, separate them with a comma (,).

Note: The attributes available for you to specify depend on the customization and configuration of your target application.

9. Click **OK**.

Selecting and installing a user registry

A user registry is required if you are using WebSphere Application Server as your point of contact server. The user registry is used as the repository for information about the users to whom you are providing single sign-on capabilities. The user registry can also be used as the repository for information about the administrative users in your environment or you can choose to keep administrative users in a separate user registry.

Before you begin

Because you are using WebSphere Application Server as your point of contact server, you can choose a user registry from many options. Refer to the WebSphere Application Server 6.1 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. Then locate information about selecting a user registry by selecting **WebSphere Application Server (Distributed platforms and Windows)** → **Securing applications and their environment** → **Authenticating users** → **Selecting a registry or repository**.

- If you are using an existing installation of WebSphere Application Server, you might have a compatible user registry already installed and configured.
- If you are using a new installation of the embedded version of WebSphere Application Server, you have several options:
  - Use the default file-based user repository realm (the federated repository), which was installed with the embedded version of WebSphere Application Server. The administrative user was configured in this registry during installation. Additional tasks needed for adding the single sign-on users are provided later in this chapter.
  - Use a different user registry. Review the WebSphere Application Server documentation for information about your user registry options. Then, install and configure the user registry you chose, if you are not using a previously existing user registry. Then configure WebSphere to use that user registry. Refer to "Configuring WebSphere to use the user registry" on page 90.

Note: If you will host your target application on a separate server, such as another WebSphere Application Server or a supported server with a plug-in such as an IHS, IIS, or Apache server, you will need to configure a user registry for that server also. Consider selecting a user registry that can be used for your point of contact server and your target application server to minimize the number of user registries you must maintain in your environment.

Configuring the user registry

The configuration of your user registry is an important step in the overall configuration.

Before you begin

Before continuing with this task, you should have already selected which user registry you will use and have installed it as described in "Selecting and installing a user registry."

About this task

In your user registry, you will create users to whom you are providing single sign-on capabilities. You can also create users for the administrators in your environment or you can choose to keep administrative users in a separate repository.

Adding single sign-on users:

In the service provider environment, the user registry is used during the creation of the local identity that is required for users to access the target application. Add these users to your user registry using the documentation for your user registry.

Adding administrative users:

If you installed the embedded version of WebSphere Application Server, a file-based user repository realm, referred to as a *federated repository* was configured for the administrative users of Tivoli Federated Identity Manager. If you would prefer to manage administrative users through the same user registry where your single sign-on users are configured, you must add them to that user registry.

Before you begin

One administrative user was created in the default user repository during the installation of Tivoli Federated Identity Manager.

About this task

To add this user to a different user registry:

Procedure

1. Create the user using the documentation for your user registry.
2. Complete the instructions in "Configuring WebSphere to use the user registry."

Configuring an SSL connection to the user registry:

After you have configured your user registry, consider enabling SSL to protect the connection between it and the server.

About this task

For instructions, refer to the WebSphere Application Server 6.1 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp. Locate information about creating SSL connections by selecting **WebSphere Application Server (Distributed platforms and Windows)** → **Securing applications and their environment** → **Securing communications**.

You might also need to refer to the documentation for your user registry.

Example

Configuring WebSphere to use the user registry

If you installed the embedded version of WebSphere Application Server, the federated repository was configured as your user registry. If you want to use a user registry other than the default federated repository, you will need to modify the WebSphere Application Server settings.

Before you begin

Before continuing with this task, review the information in "Selecting and installing the user registry" on page 72. Ensure that you have selected and installed the appropriate user registry option for your environment.

About this task

To enable WebSphere to use your user registry:

Procedure

1. Log in to the console. Select **Security** → **Secure administration, applications, and infrastructure**. The Configuration tab is displayed.
2. Click on **Security Configuration Wizard** to change the user registry used by the WebSphere runtime.
3. The **Specify extent of protection panel** is displayed. Verify that the check box **Enable application security** is selected. Click **Next**.
4. The **Secure the application serving environment** panel is displayed. Select the appropriate option for the user registry you will use:
   - **Federated repositories**
   - **Standalone LDAP registry**
   - **Local operating system**
   - **Standalone custom registry**
5. Click **Next**. The **Configure user repository** panel is displayed. Specify values for each of the registry configuration settings. Refer to the online help for descriptions of the fields presented.
6. Click **Next** and finish the wizard. Save your configuration changes.
7. Stop and then restart the WebSphere Application Server. You must use the same administrative name you used to log in and make these changes.
8. From the console, select **Tivoli Federated Identity Manager** → **Manage Configuration** → **Domain properties**.
9. In the WebSphere Security section of the panel, update the following values:

   Administrative user name

   Replace the existing entry with the LDAP administrator account name that you entered in the previous step. For example, `ldapadmin`

   Administrative user password

   Enter the password for LDAP administrator.
10. Save the changes.
11. Stop the WebSphere Application Server.
12. Restart the WebSphere Application Server.

Exporting LTPA key from the point of contact server

If you will be using your WebSphere Application Server point of contact server with a target application that is hosted by a separate WebSphere Application Server or by a server where a Tivoli Federated Identity Manager plug-in is installed, you will need to export your LTPA key so that you can share it with your target application.

Before you begin

Make sure that the date and the time settings are similar between the server from which you exported the key and the server to which you are importing the key. If the time or date is different, the server on which you will import the key might mistakenly interpret that key to be expired.

Procedure

1. Log in to the console.
2. Click **Security** → **Secure Administration, Applications, and Infrastructure** → **Authentication mechanisms and expiration**.
3. In the Password and Confirm password fields, enter the password that is used to encrypt the LTPA key. Remember the password so that you can use it later when the key is imported to the other server.
4. In the **Fully qualified key file name** field, specify the fully qualified path to the location where you want the exported LTPA key to be saved. Use the default key file name `ltpa.keys`. You must have write permission to this file.
5. Click **Export keys** to export the key to the location that you specified in the **Fully qualified key file name** field.
6. Specify the **Internal server ID** that is used for interprocess communication between servers. The server ID is protected with an LTPA token when sent remotely. By default this ID is the cell name.
7. Click **OK**.

What to do next

After exporting the key, you must share them with your target application. Refer to the appropriate instructions:

- If you are using a separate WebSphere Application Server, refer to “Importing the LTPA key to the WebSphere Application Server” on page 97.
- If you are using an Apache, IHS, or IIS server, refer to “Copying the LTPA key to the Web server” on page 100.

Chapter 10. Configuring a Web server plug-in

The Web server plug-in is required to be installed on your Web server *only* if that server is a supported server other than WebSphere Application Server. The primary function of the plug-in is to extract the user identity information from the LTPA cookie in a Web request and make the identity information available to the target application that is hosted by the Web server using either HTTP headers or server variables (if supported by the Web server).

Web request processing

In order to ensure that you can properly configure the Web server plug-in and integrate your application with the plug-in, it is helpful to understand how Web requests are processed by the plug-in.

When a request to a Web application is received by the server, it is passed to the plug-in for processing and the plug-in performs the following actions:

1. Retrieves the Web request URL.
2. Retrieves the LTPA token cookie from the request, if there is one.
3. Checks its configuration to see if the plug-in functionality is enabled. If it is not enabled, processing ends. If it is enabled, the following actions occur:
   a. Checks whether the URL in the request matches any of the URLs that are configured in the plug-in configuration file. This capability enables you to apply specific processing to specific applications.
   b. Identifies the list of HTTP headers to strip from the request. The plug-in configuration file identifies the HTTP headers to strip and prevents attacks in which "fake" headers are added by the client.
   c. Next the LTPA token cookie is examined and one of the following actions occurs:
      - If the request does not contain a valid LTPA token cookie, the plug-in identifies the list of session cookies, if any, to strip from the request based on the configuration specified in the plug-in configuration file. Cookies are stripped only if the LTPA token cookie is missing, expired, or improperly encoded. Session cookies are present only after a federated single sign-on, which is indicated by the presence of an LTPA token cookie. A session cookie without a valid LTPA token cookie implies that the session cookie is no longer applicable. Processing ends.
      - If the request contains a valid LTPA token cookie, the cookie is decoded.

        If the decoding fails or if the LTPA token has expired, no further processing occurs. The request is passed to the Web application without the addition of HTTP headers and the application is left to handle the condition.

        If the LTPA token is decoded successfully, processing continues and the plug-in creates a list of HTTP headers to set in the request. It creates a list by using the configuration specified in the plug-in configuration file and the LTPA attribute values in the token. For information about the LTPA attribute to HTTP header mapping process, see “LTPA-attribute-to-HTTP-header mapping” on page 94.

Note: Decoded LTPA tokens are saved in an in-memory cache until their expiration time. When a request is received, the plug-in checks the cache to see if a valid token is in the cache. If so, it is reused. If not, the token is decoded and added to the cache. The cache is limited in size, which is specified in number of cache entries. You can configure the size when you configure the plug-in configuration file.

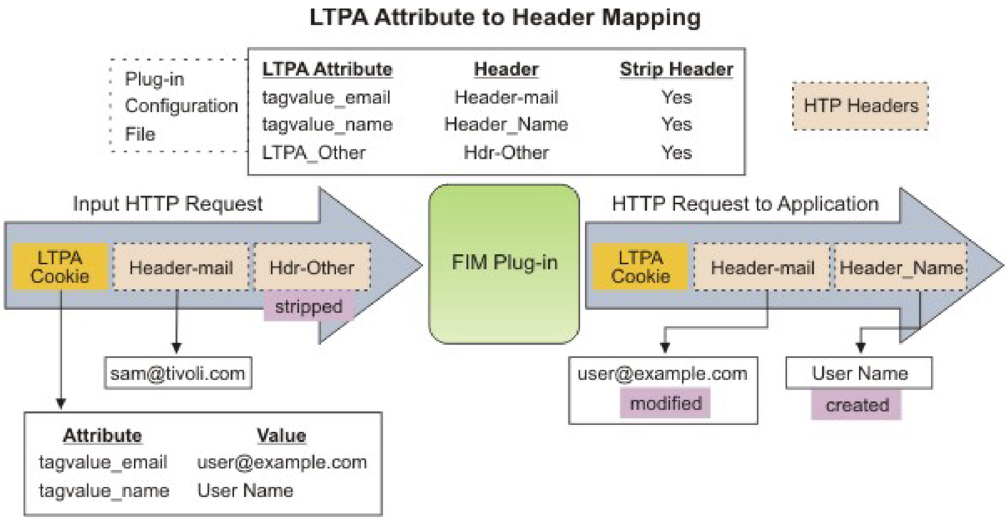
d. In the final processing step, the plug-in creates a list of server variables and values, if they are present and supported by the Web server.

   Note: The use of server variables is not supported in an IIS environment.

4. The completed Web request is then sent to the Web application to handle.

LTPA-attribute-to-HTTP-header mapping

In order to map the LTPA cookie information to an HTTP header, the plug-in relies on a special configuration file, itfimwebpi.xml, which creates and then modifies or strips (removes) the HTTP headers into the final HTTP request that is sent to the target application. The following figure shows how the content of the configuration file is used to determine the final HTTP request. Note that the figure shows only an example. LTPA attributes and headers are specific to each application that is used in an environment.



LTPA Attribute	Header	Strip Header
tagvalue_email	Header-mail	Yes
tagvalue_name	Header_Name	Yes
LTPA_Other	Hdr-Other	Yes

Attribute	Value
tagvalue_email	user@example.com
tagvalue_name	User Name

Figure 6. Example of LTPA attribute to HTTP header mapping

1. The input HTTP request in the preceding figure contains:
   - The LTPA cookie that was created by the service provider that is configured in Tivoli Federated Identity Manager
   - Two HTTP headers: 'Header-mail' and 'Other.'
2. The plug-in configuration file instructions the plug-in to map the LTPA attributes as follows:
   - LTPA attribute 'tagvalue_email' → Header-mail (strip if not in LTPA)
   - LTPA attribute 'tagvalue_name' → Header-Name (strip if not in LTPA)
   - LTPA attribute 'LTPA_Other' → Hdr-Other (strip if not in LTPA)

   For all of these headers, if the corresponding LTPA attribute does not exist, any Header with the configured name should be stripped. For example, in the figure, the LTPA value 'LTPA_Other' is not present, so the input HTTP header 'Hdr-Other' is stripped (removed). The LTPA value 'tagvalue_email' is present, so the existing header 'Header_mail' is modified to contain the value from the LTPA cookie: "user@example.com." The LTPA value 'tagvalue_name' is present, so the header 'Header_Name' is created with the value from the LTPA cookie: "User_Name." Headers that are not listed in the configuration file remain unchanged. If an LTPA cookie is not present, then all headers with "strip=yes" are removed.

The plug-in also has the ability to strip cookies if the LTPA cookie is not presented and the ability to map LTPA attributes to server variables, but these scenarios are not shown in the figure.

For information about configuring your service provider environment, including the plug-in configuration file, see “Configuring service provider components.”

Configuring service provider components

If you will be the service provider partner and are using WebSphere Application Server as your point of contact server, specific configuration tasks must be completed before you can create a federation. Additional configuration tasks are also required on the server that will host your target application.

About this task

Complete the following tasks:

1. Configure the application server that will host your target applications, as described in “Configuring your Web server.”
2. Configure your target application, as described in “Configuring the target application” on page 103.

Configuring your Web server

You have several options for the type of servers you can use to host the applications that your users can use single sign-on to access. (These applications are referred to as *target applications* because they are the target of the single sign-on request.)

About this task

Your options for servers in your Tivoli Federated Identity Manager environment include:

- IBM WebSphere Application Server 5.1 or 6.0 or later

  Note: The servers described here are usually dedicated to hosting a target application. However, you also have the option of hosting your target application on the same instance of WebSphere Application Server where you installed the Tivoli Federated Identity Manager runtime component. Your runtime component must have been installed on either of the following versions of WebSphere:
  - WebSphere Application Server version 6.1
  - The embedded version of WebSphere Application Server 6.1, which came with Tivoli Federated Identity Manager
- Microsoft Internet Information Service 6.0
- IBM HTTP Server 6.1

- Apache HTTP Server 2.0 or 2.2

When you set up your server, ensure that the Tivoli Federated Identity environment and the Web server are in the same DNS domain to enable the transfer of the LTPA cookie between the two.

To configure the Web server so that it can be used in the Tivoli Federated Identity Manager environment, complete the following tasks:

Procedure

1. Select and install a user registry for the server, as described in “Selecting and installing a user registry.”
2. Configure an SSL connection to the user registry, as described in “Configuring the user registry for the target application.”
3. “Configuring an SSL connection to the user registry” on page 97
4. If you will host a target application using a WebSphere Application Server that is separate from the server on which Tivoli Federated Identity Manager is installed, complete the steps in “Configuring a separate WebSphere Application Server to host applications” on page 97.
5. If you will host a target application using an IIS, IHS, or Apache server, complete the steps in “Configuring an IIS, IHS, or Apache server to host the application” on page 100.

Selecting and installing a user registry

A user registry is required in your environment for both your point of contact server and your application server. The users to whom you are providing single sign-on capabilities must exist in both user registries.

Before you begin

In most cases, you will want your application server to use the same user registry as the one you configured for your point of contact server. If you will use the same user registry, ensure that it is compatible with both the point of contact server and the application server.

However, if you use a separate user registry, ensure that it meets the requirements for the server that is hosting your application. Refer to your server documentation for more information. For example, if you are using WebSphere Application Server to host your application, refer to the WebSphere Application Server library and locate the information center for the version you are using: http://www.ibm.com/software/webservers/appserv/was/library/. In the appropriate information center, search for the topics about setting up a user registry.

Configuring the user registry for the target application

The configuration of your user registry is an important step in the overall configuration.

Before you begin

Before continuing with this task, you should have already selected which user registry you will use and have installed it as described in “Selecting and installing a user registry.”

About this task

If you are using the same user registry that you configured for your point of contact server, no further registry configuration is necessary. However, if you are using a separate registry, create users to whom you are providing single sign-on capabilities. These must be the same users that are defined in your point of contact user registry. Refer to the documentation for your user registry for information on adding users.

Configuring an SSL connection to the user registry

After you have configured your user registry, consider enabling SSL to protect the connection between it and the server.

About this task

If you are using the same user registry for your point of contact server and your target application server, you might have completed this task already. If you are using a separate user registry, refer to the documentation for that user registry for information about configuring SSL.

What to do next

After you have configured SSL, continue with the appropriate steps for the server on which your target applications will be hosted:

- "Configuring a separate WebSphere Application Server to host applications"
- "Configuring an IIS, IHS, or Apache server to host the application" on page 100

Configuring a separate WebSphere Application Server to host applications

In a Tivoli Federated Identity Manager environment, you can host your target applications on the same WebSphere Application Server that is used as your point of contact server or on a separate WebSphere Application Server.

About this task

To configure a separate WebSphere Application Server so that it can host applications that your single sign-on users can access, complete the following tasks:

Procedure

1. Import the LTPA key from your WebSphere Application Server point of contact server as described in"Importing the LTPA key to the WebSphere Application Server."
2. Disable the automatic generation of LTPA keys as described in "Disabling the automatic generation of an LTPA key" on page 98.
3. Configure the WebSphere Application Server to use the user registry as described in "Configuring WebSphere to use the user registry" on page 99.

Importing the LTPA key to the WebSphere Application Server

If your target application is hosted on a WebSphere Application Server that is separate from your WebSphere point of contact server, you must import the LTPA key from the point of contact server onto your target application server.

Before you begin

Before beginning this task, ensure that you have completed the following steps:

- Make sure the time between the servers is synchronized.
- Copy the LTPA keys from the location where they were exported to a location on your target application server.
- Obtain the password for the LTPA keys. A password was assigned to the keys when they were exported from the WebSphere point of contact server.

Procedure

1. Log in to the console on the *target application server*. Do not log in to your Tivoli Federated Identity Manager console to perform these steps.
2. Click **Security** → **Secure Administration, Applications, and Infrastructure** → **Authentication mechanisms and expiration**.
3. In the **Password** and **Confirm** password fields, enter the password that is used to encrypt the LTPA keys. This password must match the password that was used when the keys were exported.
4. In the **Fully qualified key file name** field, specify the fully qualified path to the location where the LTPA keys are located. You must have write permission to this file.
5. Click **Import keys** to import the keys.
6. Click **OK** and **Save** to save the changes to the master configuration.

What to do next

Next, disable the automatic generation of LTPA keys on the target application server, as described in "Disabling the automatic generation of an LTPA key."

Disabling the automatic generation of an LTPA key

By default, WebSphere Application Server automatically generates an LTPA key. However, if you are using a WebSphere Application Server other than your point of contact server to host your target application, you will use the LTPA key from your point of contact server on your application server. Therefore, you must disable the automatic key generation so that no conflicts occur.

Before you begin

To complete this task, you must know the name of the key set group and the management scope where the key set group is defined.

Procedure

1. Log in to the console on the *target application server*. Do not log in to your Tivoli Federated Identity Manager console to perform these steps.
2. Click **Security** → **SSL certificate and key management** → **Manage endpoint security configurations.**.
3. Expand the tree to the inbound or outbound management scope that contains the key set group, and then click the scope link.
4. Under Related Items, click **Key Set Groups**
5. Click the key set group that you want to disable.
6. Clear the **Automatically generate keys** check box.
7. Click **OK** and **Save** to save the changes to the master configuration.

What to do next

Continue with the steps for "Configuring WebSphere to use the user registry."

Configuring WebSphere to use the user registry

Ensure that the WebSphere Application Server that you are using to host your target application is configured to use the user registry that you selected and installed.

Before you begin

Before continuing with this task, review the information in "Selecting and installing the user registry" on page 72. Ensure that you have selected and installed the appropriate user registry option for your environment.

About this task

To enable WebSphere to use your user registry:

Procedure

1. Log in to the console *for your target application*. Do not log in to your Tivoli Federated Identity Manager console to perform these steps.
2. Select **Security** → **Secure administration, applications, and infrastructure**. The Configuration tab is displayed.
3. Click on **Security Configuration Wizard** to change the user registry used by the WebSphere runtime.
4. The **Specify extent of protection panel** is displayed. Verify that the check box **Enable application security** is selected. Click **Next**.
5. The **Secure the application serving environment** panel is displayed. Select the appropriate option for the user registry you will use:
   - **Federated repositories**
   - **Standalone LDAP registry**
   - **Local operating system**
   - **Standalone custom registry**
6. Click **Next**. The **Configure user repository** panel is displayed. Specify values for each of the registry configuration settings. Refer to the online help for descriptions of the fields presented.
7. Click **Next** and finish the wizard. Save your configuration changes.
8. Stop and then restart the WebSphere Application Server. You must use the same administrative name you used to log in and make these changes.
9. From the console, select **Tivoli Federated Identity Manager** → **Manage Configuration** → **Domain properties**.
10. In the WebSphere Security section of the panel, update the following values:

    Administrative user name
    Replace the existing entry with the LDAP administrator account name that you entered in the previous step. For example, `ldapadmin`

    Administrative user password
    Enter the password for LDAP administrator.
11. Save the changes.
12. Stop the WebSphere Application Server.

13. Restart the WebSphere Application Server.

What to do next

When you are done, continue with the appropriate step for your environment:

- If you will be hosting applications on an IHS, IIS, or Apache server, continue with "Configuring an IIS, IHS, or Apache server to host the application."
- If you will hosting applications on only your WebSphere server, your server configuration is complete. Continue with the target application configuration in "Configuring the target application" on page 103.

Configuring an IIS, IHS, or Apache server to host the application

If you will host your target applications using a Microsoft Internet Information Services server, an IBM HTTP Server, or an Apache HTTP Server, you must complete specific configuration tasks.

Before you begin

Before continuing with these tasks, you must have installed the plug-in.

Ensure that:

- The plug-in is installed on the server that is hosting the target application.
- The server is in the same domain as the Tivoli Federated Identity Manager server.

Also, ensure that you have completed the steps in "Configuring your Web server" on page 95, including:

- "Selecting and installing a user registry" on page 96
- "Configuring the user registry for the target application" on page 96
- "Configuring an SSL connection to the user registry" on page 97

About this task

Complete the following tasks to prepare your plug-in environment:

Procedure

1. Copy the LTPA key to your server, as described in "Copying the LTPA key to the Web server."
2. Create the plug-in configuration file, as described in "Creating the plug-in configuration file" on page 101.
3. Copy the plug-in configuration file to the server, as described in "Copying the plug-in configuration to the server" on page 102.

Copying the LTPA key to the Web server

The LTPA key that is used by WebSphere Application Server on your point of contact server must be shared with the server where the plug-in is installed.

Before you begin

Before you continue with the steps for copying the LTPA key to your server, ensure that you have completed the following tasks:

- Installed the plug-in on the Web server.
- Completed the configuration of your point of contact server, as described in "Configuring a WebSphere Application Server point of contact server (service provider)" on page 87.
- Exported the LTPA keys from your point of contact server, as described in "Exporting LTPA key from the point of contact server" on page 91.
- Verified that the time on the point of contact server and on the server to which you are copying the LTPA key are synchronized.

Procedure

1. Copy the LTPA key, which should be named `ltpa.keys`, from the location to which it was exported.
2. Paste the LTPA key in the webpi directory on the application server. For example:

   On an IHS or Apache server:
   ```
   /opt/IBM/FIM/webpi/etc
   ```

   On an IIS server:
   ```
   C:\Program Files\IBM\FIM\webpi\etc
   ```

What to do next

Continue with "Creating the plug-in configuration file."

Creating the plug-in configuration file

After you have installed the plug-in and prepared your environment to use the plug-in, you must configure it with specific information about the Web applications that will be accessed by your single sign-on users.

Before you begin

To complete this task, you will need the following information:

- The password that was used to encrypt the LTPA key when it was exported.
- The name and URL of each target application that is hosted by this server.
- The appropriate HTTP header and LTPA attribute mappings for your environment. You must know which LTPA attribute you want to map to which HTTP header or server variable. The HTTP header and server variables are those expected by the target application.
- A list of cookies to remove if the LTPA cookie is missing or is not valid, which usually indicates that the user is not a federated single sign-on user.
- A list of mappings between server variable names and LTPA token attribute names. Server variables are an alternative mechanism for presenting LTPA attributes to the application instead of using HTTP headers.

  Note: The use of server variables is not supported on IIS.

For more information about HTTP header and LTPA attribute mappings and how the plug-in functions in the environment, refer to Chapter 10, "Configuring a Web server plug-in," on page 93.

Procedure

1. Log in to the console.

2. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Web Server Plugin Configuration**. The Web Plugin Single Sign-on Configuration panel is displayed.
3. Complete the information that is required for your server in the **Web Server Plug-in Single Sign-on Configuration** and the **Web Server Plug-in Logging Configuration** sections. Refer to the online help for descriptions of the fields. **Note:** Make sure the that the password you specify in the LTPA password field matches the password you created when you exported the ltpa.keys file. When you have completed all of the fields, click **Save**.
4. In the **Web Server Plug-in Applications Configuration**, define an application to the single sign-on configuration by clicking **Create**. The Application Properties panel is displayed.
   a. Complete the information about the application that you want to make available to your single sign-on users.
   b. Click **Apply**.
   c. Click **HTTP Header to LTPA Attribute Mappings**.
   d. Accept the default settings by clicking **Apply** or modify the settings by clicking **Create**.
   e. When you have completed this panel, click **Apply**.
   f. Click **Client Cookies to be Removed.**
   g. Accept the default settings by clicking **Apply** or modify the settings by clicking **Create**.
   h. When you have completed this panel, click **Apply**.
   i. Click **Server Variables to LTPA Attribute Mappings**.
   j. Accept the default settings by clicking **Apply** or modify the settings by clicking **Create**.
   k. When you have completed this panel, take one of the following actions:
      - If you want to add other applications, click **Apply** and then repeat the preceding steps for each application until all additional applications have been added.
      - If you have completed the addition of the application to the server, click **OK**.
5. Click **Save**.
6. Click **Export Web Server Plug-in Configuration File**. Then complete the following steps:
   a. Click **Save** in the pop-up window to save the configuration to a file called `itfimwebpi.xml`.
   b. Select the installation directory for your Web server plug-in. For example, save `itfimwebpi.xml` to the `/opt/IBM/FIM/webpi/etc` directory.

What to do next

Continue with “Copying the plug-in configuration to the server.”

Copying the plug-in configuration to the server

After you have created the plug-in configuration file, you must copy that configuration to your Web server.

Procedure

1. Locate the configuration file that you created using the steps in “Creating the plug-in configuration file” on page 101. The file is named `itfimwebpi.xml` and was created in the directory that you specified when you exported the file.
2. Copy the file and then paste the file in the webpi directory on your Web server:

   On an IHS or Apache server:
   ```
   /opt/IBM/FIM/webpi/etc
   ```

   On an IIS server:
   ```
   C:\Program Files\IBM\FIM\webpi\etc
   ```
3. Restart your Web server for the changes to take effect.

What to do next

The configuration of your server is complete. Continue with the target application configuration in “Configuring the target application.”

Verifying plug-in configuration on Apache or IBM HTTP Server

After you have configured the plug-in on an Apache HTTP Server or an IBM HTTP Server, you can verify that the configuration was successful.

Before you begin

Before continuing with this task, ensure that you have completed the following tasks:

- “Configuring a WebSphere Application Server point of contact server (service provider)” on page 87
- “Configuring an IIS, IHS, or Apache server to host the application” on page 100

Procedure

1. On the server, locate the httpd.conf file. The location of this file is dependent on your installation. For example:
   ```
   /etc/httpd/conf/httpd.conf
   ```
2. Open the file in a text editor and locate the appropriate line for the plug-in you are using:

   Apache HTTP Server 2.2:
   ```
   LoadModule fimwebpi_module /opt/IBM/FIM/webpi/lib/libitfimwebpi-apache22.so
   ```

   Apache HTTP Server 2.0 or IBM HTTP Server:
   ```
   LoadModule fimwebpi_module /opt/IBM/FIM/webpi/lib/libitfimwebpi-apache20.so
   ```

   Make sure that the webpi module (`libitfimwebpi-apache22.so` or `libitfimwebpi-apache20.so`) has write access to the log file path that is defined in your plug-in configuration file (`itfimwebpi.xml`).

What to do next

Continue with the tasks in “Configuring the target application.”

Configuring the target application

As the service provider, your role in the federation is to provide a service, such as a Web application, to the user.

About this task

As part of this role, you will need to ensure that the application (referred to as the *target application*) that you are providing to the users is configured appropriately for use in a Tivoli Federated Identity Manager environment:

- The application must be able to accept user identity information using an HTTP headers or server variables.
- The Tivoli Federated Identity Manager environment and the application must be in the same DNS domain.
- The application must be hosted by a supported Web server, such as:
  - Microsoft Internet Information Services (IIS) server 6.0, with the Tivoli Federated Identity Manager plug-in installed
  - IBM HTTP Server 6.1, with the Tivoli Federated Identity Manager plug-in installed
  - Apache HTTP Server 2.0 or 2.2, with the Tivoli Federated Identity Manager plug-in installed
  - WebSphere Application Server version 5.1
  - WebSphere Application Server version 6.0 or later

    Note: You can also use the same instance of WebSphere Application Server where you installed the Tivoli Federated Identity Manager runtime component as the server to host your target application. The version of that WebSphere Application Server is either:

    - WebSphere Application Server version 6.1 with fix pack 15
    - The embedded version of WebSphere Application Server, which came with Tivoli Federated Identity Manager

For information about configuring your target application, refer to the documentation for the server that will host the application. For example, if you are hosting your target application on WebSphere Application Server, refer to the Information Center for the version of WebSphere Application Server you are using from the library at http://www.ibm.com/software/webservers/appserv/was/library/.

Configuring the login for your application

Prior to using Tivoli Federated Identity Manager, you probably used a login method that was specific to your application. For example, you might have provided a URL to your users that directed them to a login form or required client authentication. In your Tivoli Federated Identity Manager environment, your identity provider partner will be responsible for authenticating users. Therefore, depending on the configuration of your federation, you will likely need to either direct your users to a new URL (such as one hosted by your identity provider partner) or by redirecting users from your site to the appropriate login method used by your identity provider partner.

About this task

Discuss login requirements with your identity provider partner. Then ensure that your environment is configured appropriately to send your users to the appropriate login location.

Instructing users to enable cookies

Users must be sure to enable cookies in their browsers when using single sign-on to a service provider who is using WebSphere Application Server as its point of contact server.

About this task

Advise users to follow the instructions for enabling cookies for their browsers.

Chapter 11. Setting up the alias service database

SAML 2.0 supports the use of name identifiers (aliases) for communication of user identities between partners. Aliases are intended to increase the privacy of the user when that user accesses resources at a service provider. When aliases are used, an identifier that both the identity and service provider will recognize is sent instead of the user's actual account name. Aliases are created and recorded during account linkage (federation). After account linkage, the alias is in all messages that are sent between the partners. A different alias is used with each partner. Also, the alias used in one direction (such as from identity provider to service provider) can be different from the alias that is used in the other direction (such as from service provider to identity provider). The use of aliases is optional in SAML 2.0.

About this task

The default setting for the alias service is to use persistent IDs.

A service in Tivoli Federated Identity Manager, called the *alias service*, generates new aliases, associates aliases with local users, and performs mapping from alias to user and from user to alias.

Most aliases are persistent and must be retained for a long period of time. Therefore, some type of database must be used to store them. You have two options for the type of database you can use:

- JDBC database, such as the Derby database in WebSphere Application Server
- LDAP database, such as IBM Tivoli Directory Server, which is available separately

The tasks you must perform to set up your alias service database depend on whether you installed the embedded version of WebSphere Application Server or are using an existing version of WebSphere Application Server with your installation of the Tivoli Federated Identity Manager Runtime and Management Services component.

Embedded version of WebSphere

Your database options are:

- **JDBC database**

  If you installed the embedded version of WebSphere Application Server, a JDBC database (Cloudscape 10, also known as Derby) was configured on WebSphere Application Server to be used for storing alias information. No further tasks for setting up the database are required.
- **LDAP database**

  You have the option of using an LDAP database, such as IBM Tivoli Directory Server, that you have purchased, installed, and configured separately from Tivoli Federated Identity Manager. Refer to the information in “Configuring an LDAP alias service database” on page 110. Then, to use that LDAP database with Tivoli Federated Identity Manager, you must modify the alias service settings, as described in “Modifying alias service settings” on page 109.

Existing version of WebSphere Application Server

Your database options are:

- **JDBC database**

  If you installed Tivoli Federated Identity Manager on an existing version of WebSphere Application Server and you want to use a JDBC database, you must manually create and configure the database, using a procedure similar to those described below for Cloudscape 10 (Derby), as described in "Configuring a JDBC alias service database." (As previously stated, if you installed the embedded version of WebSphere Application Server, these steps were performed automatically and are already completed.)
- **LDAP database**

  You have the option of using an LDAP database, such as IBM Tivoli Directory Server, that you have downloaded, installed, and configured separately from your Tivoli Federated Identity Manager. Refer to the information in "Configuring an LDAP alias service database" on page 110. Then, to use that LDAP database with Tivoli Federated Identity Manager, you must modify the alias service settings, as described in "Modifying alias service settings" on page 109.

Configuring a JDBC alias service database

If you installed Tivoli Federated Identity Manager on an existing version of WebSphere Application Server and you want to use a JDBC database, you must manually create and configure the database using the procedure that follows. If you installed the embedded version of WebSphere Application Server, these steps were performed automatically and are already completed.

About this task

The following instructions describe how to create and use the JDBC Derby database that is provided with WebSphere Application Server. The Derby database is created by an Apache tool called ij. It is implemented with the Java class org.apache.derby.tools.ij.

Procedure

1. Create the FIMAliases database and import the schema:
   a. Open a command prompt and start the ij tool, which is located in the /derby/bin/embedded directory where you installed WebSphere Application Server.

      On AIX, HP-UX, Linux, or Solaris, type `$was_home/derby/bin/embedded/ij.sh`.

      On Windows, type `$was_home/derby/bin/embedded/ij.bat`.

   b. From the ij command line, create the database and the schema by typing the following commands:

      ```
      connect 'jdbc:derby:FIMAliases;create=true';
      run '/opt/IBM/FIM/etc/Table.ddl';
      quit;
      ```

      Note: The location of the `Table.ddl` file is in the installation directory of Tivoli Federated Identity Manager; If you used a different installation directory, use that path with the run command. On Windows, the default installation directory is `C:\Program Files\IBM\FIM`.
2. Verify the database and schema:
   a. Open a command prompt and locate the *$was_home*/derby/FIMAliases directory.

b. Verify that the output SQL file contains the FIMAliases schema.

   On AIX, HP-UX, Linux, or Solaris, type:

```
$was_home/derby/bin/embedded/dblook.sh -d jdbc:derby:FIMAliases -o FIMAliase.sql
```

   On Windows, type:

```
$was_home/derby/bin/embedded/dblook.bat -d jdbc:derby:FIMAliases -o FIMAliases.sql
```

3. Create the Derby embedded JDBC provider and data source:
   a. Open the WebSphere administration console and click **Resources** → **JDBC** → **JDBC Providers**.
   b. Click **New**.
   c. Complete the required fields as follows:

      Database type
         Select **Derby**.

      Provider type
         Select **Derby JDBC Provider**.

      Implementation type
         Select **Connection pool data source**.

      Name Use a name to indicate that this is the JDBC provider of the alias service for Tivoli Federated Identity Manager. For example, use ITFIM Alias Service JDBC Provider.
   d. Then click **Next** and then **Finish**.
4. Create a data source for this JDBC provider:
   a. In the WebSphere administration console and click **Resources** → **JDBC** → **JDBC Providers** → **ITFIM Alias Service JDBC Provider** → **Data sources** → **New**.
   b. Complete the required fields as follows:

      Database source name
         Type a name that identifies the datasource, such as `ITFIM Alias Service Datasource`.

      JNDI name
         Type `jdbc/IdServiceJdbc`.

         Attention: Use this name exactly as shown so that the mappings between the alias service and the data source will occur automatically.
   c. Click **Next**.
   d. Provide a name for the database, such as `FIMAliases`.
   e. Click **Next** and click **Finish**.
5. To verify the connection to the database, Select the datasource you configured and click **Test connection**.

Modifying alias service settings

About this task

To modify the setting for your name identifier database:

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Alias Service Settings**. The Alias Service Settings portlet is displayed.
2. Select **JDBC Provider and Data Source**

   Use this option if you will use a JDBC database to store name identifier information.
3. Click **Apply** and then click **OK**.

Configuring an LDAP alias service database

If you install Tivoli Federated Identity Manager with embedded WebSphere, a JDBC database is the default setting for an alias service database in Tivoli Federated Identity Manager. However, you can choose to use an LDAP database instead.

Before you begin

If you install Tivoli Federated Identity Manager with an existing WebSphere deployment, you might already have an LDAP database in use as a user registry. When using WebSEAL as the point of contact server, you are installing into an environment that includes Tivoli Access Manager. The most common LDAP deployment with Tivoli Access Manager is IBM Tivoli Directory Server.

The Tivoli Federated Identity Manager alias service stores alias information in a user registry. The alias service supports the following user registries:

- IBM Tivoli Directory Server
- Sun ONE

Note:

You can write your own alias service for use with other registries, such as Lotus® Domino® or Microsoft Active Directory.

The alias service requires a location in LDAP to store the necessary information, and the Tivoli Federated Identity Manager runtime and management services feature needs an account on the LDAP server to search for alias information.

If you do not have an LDAP database installed already, you must install an LDAP product in order to use the alias service.

If you need LDAP, you could use the IBM Tivoli Directory Server product, which can be downloaded from http://www-306.ibm.com/software/tivoli/resource-center/security/code-directory-server.jsp.

About this task

If you use an LDAP database, the following configuration tasks are required:

- "Using tfimcfg to configure LDAP for the alias service" on page 111

  Tivoli Federated Identity Manager provides a utility that automates this process, when used with either IBM Tivoli Directory Server or Sun ONE Directory Server.Configuring the LDAP user registry for the alias service
- "Creating an LDAP suffix" on page 114

- “Modifying alias service settings” on page 109

Using tfimcfg to configure LDAP for the alias service

About this task

You can use the tfimcfg utility to automate the LDAP configuration for the alias service. This installation guide will instruct you to run tfimcfg for the purpose of configuring the alias service.

The tfimcfg uses a data file called ldapconfig.properties to decide which actions to take. You can modify the tfimcfg behavior by using a text editor to modify values in this file. You can specify whether or not specific sets of LDAP properties will be defined. For each set that you choose to create, you can specify the values of the individual properties.

In order for tfimcfg to configure LDAP programmatically, the utility must know some LDAP information, such as the LDAP hostname, LDAP port number, and administrator account information. The ldapconfig.properties file contains entries for each of these properties. Default values are provided. You will need to modify the values to fit your deployment environment.

The following steps list the properties for which you should define a value.

Procedure

1. Obtain a copy of ldapconfig.properties. You can view the file contents by either:
   - Viewing the default file listing in Appendix A, “tfimcfg reference,” on page 491
   - Accessing your installation software (CD or installation directory), and viewing the default file:

     AIX, Solaris, HP-UX, or Linux

     ```
     /opt/IBM/FIM/tools/tamcfg/ldapconfig.properties
     ```

     Windows

     ```
     C:\Progra~1\IBM\FIM\tools\tamcfg\ldapconfig.properties
     ```

     z/OS

     ```
     /usr/lpp/FIM/tools/tamcfg/ldapconfig.properties
     ```
2. Specify whether tfimcfg adds suffixes to the LDAP server as needed.

   Default:

   ```
   ldap.suffix.add=true
   ```

   The tfimcfg utility adds a number of suffixes, based on other settings in the ldapconfig.properties file. If you want to override the creation of any suffixes, set this value to false.

   The suffixes that can be created are:
   - A suffix for the hierarchy to hold alias service information (user identity aliases)

     Default: cn=itfim
   - A suffix for use by the Tivoli Access Manager servers

     Default: secAuthority=Default
   - A suffix for a hierarchy for storing user and group information

     Default: dc=com

3. Specify whether tfimcfg creates LDAP containers to store Tivoli Federated Identity Manager server users and groups.

   The Tivoli Federated Identity Manager users and groups are:

   Default:

   ```
   ldap.suffix.user.configuration=true
   ldap.organization.configuration=true
   ```

   - When `ldap.suffix.user.configuration=true`, tfimcfg adds an LDAP suffix `dc=com` and creates an object for it. The utility also sets additional properties as specified in ldapconfig.properties. The list of properties, with their default values, is:

     ```
     ldap.suffix.user.dn=dc=com
     ldap.suffix.user.name=com
     ldap.suffix.user.attributes=dc
     ldap.suffix.user.objectclasses=domain
     ```

   - When `ldap.organization.configuration=true`, tfimcfg sets additional properties. The properties are specified in ldapconfig.properties. The list of properties, with their default values, is:

     ```
     ldap.user.container.dn=cn=users,dc=example,dc=com
     ldap.group.container.dn=cn=groups,dc=example,dc=com
     ldap.organization.dn=dc=example,dc=com
     ldap.organization.name=example
     ldap.organization.attributes=dc
     ldap.organization.objectclasses=domain
     ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
     ldap.group.objectclasses=groupOfUniqueNames
     ldap.user.shortname.attributes=cn,sn,uid
     ```

   You can use a text editor to modify the values for these LDAP containers.

4. Specify whether tfimcfg creates an LDAP suffix to store single sign-on aliases.

   Default:

   ```
   ldap.suffix.alias.configuration=true
   ```

   When you do not want the utility to specify a new suffix, set this to `false`.

   When this property is set to `true`, tfimcfg uses the value set in the following property:

   ```
   ldap.suffix.alias.dn=cn=itfim
   ```

   You can use a text editor to modify the DN value. This value of this property must begin with `cn=`.

5. Specify whether tfimcfg creates the `secAuthority=Default` suffix for Tivoli Access Manager.

   This suffix is used by Tivoli Access Manager to define an LDAP hierarchy for use by the Tivoli Access Manager servers. This suffix is typically created by the Tivoli Access Manager installation scripts. The tfimcfg utility adds the suffix if it does not already exist.

   Default:

   ```
   ldap.suffix.tam.configuration=true
   ```

   - When you have already configured Tivoli Access Manager set this value to `false`.
   - When Tivoli Access Manager is not using this LDAP server, set this value to `false`.

   Note: When the secAuthority=Default suffix already exists, the tfimcfg program ignores the value of the ldap.suffix.tam.configuration property.

6. Specify whether tfimcfg configures LDAP for the Tivoli Federated Identity Manager alias service.

Default:

```
ldap.fim.configuration=true
```

Default value: `true`.

When this value is true, tfimcfg sets the following properties, as specified in ldapconfig.properties:

- The distinguished name, short name, and password that the Tivoli Federated Identity Manager server (runtime and management service) uses to bind to the LDAP server. Defaults:

  ```
  ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
  ldap.fim.server.bind.shortname=fimserver
  ldap.fim.server.bind.password=passw0rd
  ```

- The distinguished name and short name for the group to which the user identity for the Tivoli Federated Identity Manager server (fimserver) belongs. Defaults:

  ```
  ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
  ldap.fim.admin.group.shortname=fimadmins
  ```

The tfimcfg utility then adds the user:

```
uid=fimserver,cn=users,dc=example,dc=com
```

to the group:

```
cn=fimadmins,cn=groups,dc=example,dc=com
```

7. Specify whether tfimcfg attaches appropriate ACLs (access control lists) to the LDAP server.

   Default:

   ```
   ldap.modify.acls=true
   ```

   When this is set to `false`, you must attach the ACLs manually.

   These ACLs grant read and write access to the Tivoli Federated Identity Manager administrative users created by tfimcfg.

   For example, when `ldap.modify.acls=true`, and tfimcfg is run using the default values for creation of suffixes, ACLs are set for the following suffixes:

   - cn=itfim
   - secAuthority=Default
   - dc=com

   Note: tfimcfg attaches ACLs for IBM LDAP and Sun ONE servers. For other LDAP servers, you must attach the ACLs manually.

8. Specify values for each property that describes your LDAP deployment.

   Default values are provided for most properties. Modify the properties to match your deployment. When LDAP security is enabled, enter the name of the Java keystore that contains the certificate used for SSL, and enter the password to be used by the Tivoli Federated Identity Manager management service.

Table 12. LDAP properties to modify for tfimcfg

Property	Description	Your value
ldap.hostname	The system that hosts the LDAP server. Default value is `localhost`.	
ldap.port	The LDAP port. Default value is `389` for non-SSL communication.	
ldap.admin.dn	LDAP administrator name. Default: `cn=root`	
ldap.admin.password	The password for the LDAP administrator	

Table 12. LDAP properties to modify for tfimcfg (continued)

Property	Description	Your value
ldap.security.enabled	Boolean value that specifies whether LDAP security is enabled. This value is disabled by default.	
ldap.security.trusted.jks.filename	The name of the Java keystore that contains the signer of the LDAP-presented SSL certificate that LDAP presents during trusted communications. There is no default entry.	
ldap.fim.server.bind.password	The password for servers that communicate with the LDAP servers. You will want to change the default to values used in your deployment.	

9. To configure the LDAP server, see Appendix A, "tfimcfg reference," on page 491.

Creating an LDAP suffix

You must create an LDAP suffix, such as `cn=itfim`, to enable the alias service to access the LDAP user registry.

Before you begin

The following instructions are for IBM Tivoli Directory Server. Make sure you have installed IBM Tivoli Directory Server and completed initial configuration as described in its documentation before continuing with the following steps.

Procedure

1. Stop the IBM LDAP server.

   AIX, HP-UX, Linux, or Solaris:
   ```
   # ibmdirctl -D cn=root -w passw0rd stop
   ```

   Windows
   Use the Services icon.
2. Add the suffix: `# idscfgsuf -s "cn=itfim"`.
3. Start the IBM LDAP server.

   AIX, HP-UX, Linux, and Solaris:
   ```
   # ibmdirctl -D cn=root -w passw0rd start
   ```

   Windows
   Use the Services icon.
4. Use **ldapmodify** to update the LDAP schema file. For example, on Linux:
   ```
   ldapmodify -D cn=root -w passw0rd -f
     /opt/IBM/FIM/etc/itfim-secuser.ldif
   ```

Planning configuration of the alias service properties

Use these instructions to specify the alias service properties for accessing one or more LDAP servers.

About this task

The alias service manages aliases by accessing an LDAP user registry. The alias service needs to know a number of pieces of information about the LDAP environment in which it will operate. The management console provides a GUI interface that you can use to specify the necessary properties. The properties are stored in a Tivoli Federated Identity Manager property file specific to the current Tivoli Federated Identity Manager domain.

This topic describes the properties that you will need to specify, and provides a worksheet that you can use to enter the values for your environment. For many properties, you will be able to use a default value.

The value to set for some of the properties will correspond to values that you specified previously, when you planned the use of the tfimcfg utility. At that time, you identified values to edit in the file ldapconfig.properties. The tables in the following task sequence identify the GUI fields with values that should match the properties in ldapconfig.properties.

Procedure

1. Determine the value for the LDAP search property.

   The following table describes the Root suffix, the LDAP search property that is configurable through the GUI. You can expedite the configuration by identifying at this time the appropriate value for your deployment environment.

Table 13. LDAP Search property

Property	**Description**	**Your value**
Root suffix	Specifies the root suffix where alias settings are written. This property can have one value (suffix) only. The value of this property matches the value for the following property in ldapconfig.properties: `ldap.suffix.alias.dn` For example: `cn=itfim`.	

2. Determine values for LDAP environment properties.

Table 14. LDAP environment properties

Property	**Description**	**Your value**
SSL Enabled	A check box that specifies whether communication between the alias service and the LDAP servers should be secured using Secure Socket Layer (SSL). When the LDAP servers are configured to use SSL, the alias service must use SSL when communicating with them. This value of this property corresponds to the value for the following property in ldapconfig.properties: `ldap.security.enabled` When using SSL, the **SSL Enabled** check box should be selected, and the value of ldap.security.enabled should be `true`.	
Keystore	When the **SSL Enabled** check box is selected, you must select a keystore from the **Keystore** menu list. The selected keystore is the name of the trusted keystore containing the CA certificate of the LDAP server. Note that the certificate authority certificates for all LDAP servers must be in the same keystore. This value of this property corresponds to the value for the following property in ldapconfig.properties: `ldap.security.trusted.jks.filename`	

3. Determine values for LDAP server properties

Table 15. LDAP server properties

Property	Description	Your value
LDAP Hostname	The **LDAP Hosts** box lists the configured servers in order of preference. The alias service tries first to contact the server at the start (top) of the list. If that contact is unsuccessful, the alias service attempts to contact the next server on the list. This value of this property includes the value for the following property in ldapconfig.properties: `ldap.hostname` The ldapconfig.properties file holds only one value for this property, but you can specify multtiple values for LDAP Hostname.	
Port	The port on which the LDAP server listens. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.port` Default port for non-SSL communication: `389` Default port for SSL communication: `636`	
Bind DN	The distinguished name (DN) that the alias service uses to bind to the LDAP server. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.fim.server.bind.dn` The GUI panel provides a default of cn=root. However, root access is not required in order to complete the bind. You can specify the DN of the alias service. Default value: `uid=fimserver`	
Bind Password	The password for the DN specified in the **Bind DN** field. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.fim.server.bind.password`	
Mode	The default is read-write. When you configure multiple LDAP servers, typically only one should be read-write. In this scenario, the other LDAP servers are typically deployed for failover purposes, and are expected to have read-only copies of the user registry.	
Minimum number of connections	The initial number of connections (binds) for the alias service to establish to the LDAP server. The minimum valid number is zero (0). The maximum valid number is limited only by the maximum value supported by the data type. The default value is 2. Use the default value unless you have a specific need to increase it.	
Maximum number of connections	The maximum number of connections (binds) for the alias service to establish to the LDAP server. The maximum valid number is limited only by the maximum value supported by the data type. The default value is 10. Use the default value unless you have a specific need to increase it.	

Modifying alias service settings for LDAP

About this task

To modify the setting for your name identifier database:

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Alias Service Settings**. The Alias Service Settings portlet is displayed.
2. Select **LDAP**.

   Specify the properties that you put in the worksheet in "Planning configuration of the alias service properties" on page 114
3. If you select SSL for communication with the LDAP, you need to select the name of the trusted keystore containing the CA of the LDAP server. If you have not already moved the LDAP CA to the Tivoli Federated Manager key service, you can retrieve the certificate over SSL as follows:
   a. In the console, click **Tivoli Federated Identity Manager** → **Key Service**.
   b. Select the truststore where you want to store the certificate in the Keystore table. The View Keys button is activated.
   c. Click **Retrieve Certificate from SSL Connection**. The Password panel is displayed.
   d. Type your truststore password and click **OK**.
   e. Complete the fields to specify the host name and port name from which you will retrieve the certificate. Optionally, click the Show Signer Info to view the certificate before retrieving.
   f. Complete the Alias field with the name you want to use for the certificate. Then, click **OK**. The certificate is added to the truststore.
4. Click **Apply** and then click **OK**.

Chapter 12. Planning the mapping of user identities

Task overview:

1. Read this series of topics on the mapping of user identities
2. Review the default mapping rules files for your protocol. Decide if you can use them, either as they are, or by making your own modifications as appropriate for your deployment.
3. If the requirements for your deployment cannot be met by the use of a mapping rule, you can choose one of the following options:
   - Use the Tivoli Directory Integrator mapping module that is provided withTivoli Federated Identity Manager.
   - Develop a custom mapping module.

A primary function of the Tivoli Federated Identity Manager trust service is the transfer of user identity information (credentials) between partners in a single sign-on federation. This transfer requires that user identity information change formats several times, in order to move between formats local to each partner and the token format that has been agreed to for exchanging credentials.

Part of this transfer of identity information is an identity mapping step, in which user information is mapped from the structure provided by one credential or token type into the structure required by another token type.

To complete this mapping step, you must choose one of the following options:

- Write an identity mapping rule
- Deploy the Tivoli Directory Integrator mapping module

  Use of this module requires an understanding of the Tivoli Directory Integrator features and configuration. You will want to consult the product documentation for Tivoli Directory Integrator.

  Tivoli Federated Identity Manager provides a GUI interface for setting some configuration properties. See “Tivoli Directory Integrator identity mapping module” on page 126.
- Develop a custom mapping module.

  This is a development task, to build your own module that is tailored to the needs of the applications in your deployment. See “Creating a custom mapping module” on page 134.

If you choose to write an identity mapping rule, you will use the eXtensible Stylesheet Language (XSL), and save it to disk as an XSL file. When you create a federation, the federation wizard will prompt you to supply the name of your mapping rule file. The wizard will import this file into the configuration for the federation.

Each identity mapping rule file is specific to a particular role and a particular federation. For example, when you create a SAML federation for an identity provider, the mapping rule you must use is different from the rule you will use when you create a SAML federation for a service provider. Also the identity mapping rule for a Liberty federation on an identity provider is different from the mapping rule for a SAML federation on an identity provider.

This means that before you create a federation, you must create and save a mapping rule file.

Note: An identity mapping rule specifies the attributes that are associated with a user's credential. Users can access multiple applications after they authenticate, so you need to make sure that your rule sets the appropriate attributes for all of the applications that the user accesses.

The Tivoli Federated Identity Manager management console provides a Federation wizard that guides you through the configuration of a single sign-on federation. The Wizard contains an Identity Mapping panel, which prompts the administrator to supply the name of an identity mapping rule file. The wizard imports the file, and uses it when building the configuration for the trust module chain that is specific to the federation.

The administrator must create the identity mapping file prior to using the wizard to configure the federation. The wizard panel expects that the administrator has created an eXtensible Stylesheet Language (XSL) file that describes identity mapping rules. The identity mapping rules are used to convert information that must move across the federation between the partners (identity provider and service provider). Each identity mapping rule must provide:

- The information structure that is required by the security token to be generated.
- The information content (identity attributes) that is required by applications that use the federation.

In order to write an identity mapping rule, you need to understand:

- The role of the identity mapping module.
- The expression of user identity information in XML files.
- The use of the XSL language to specify rules for manipulating the user identity information.

Identity mapping overview

When exchanging security tokens with partners, it is not enough to simply understand the different token standards. It is just as important to know what information a particular partner is expecting in tokens from your site, and what information you should expect to receive from partners. The Tivoli Federated Identity Manager identity mapping and trust service functions enable you to customize the format and content of incoming and outgoing tokens to meet each partner's requirements.

In a single sign-on federation, an identity provider is responsible for authentication of the end user, resulting in the creation of user credentials in the identity provider environment. For example, an identity provider might require users to authenticate with a user name and password. The user information is validated against the identity provider's user registry, and a local credential is created that contains group membership data along with optional attributes about the user. In the most typical use case of SAML 2.0, the username is not carried by the assertion. Instead, the user is represented by an alias.

A service provider will also have specific requirements for its users' credentials, which are needed for users to access applications. In many cases the credentials required by the service provider will differ in format or content from the credentials created by the identity provider. For example, the service provider might want a specific attribute to be included in the credential, such as the user's social security number. Therefore, the identity credentials might need to be mapped between the identity provider and the service provider. In Tivoli Federated Identity Manager, on the identity provider side, the locally authenticated user (the input identity) can be mapped to a different user before the creation of the single sign-on token (the output identity). Similarly, on the service provider side, the identity that is received from the sign-on token (the input identity) can be mapped to a local identity that is needed for access to service provider applications (the output identity). The mapping process is shown in Figure 7.

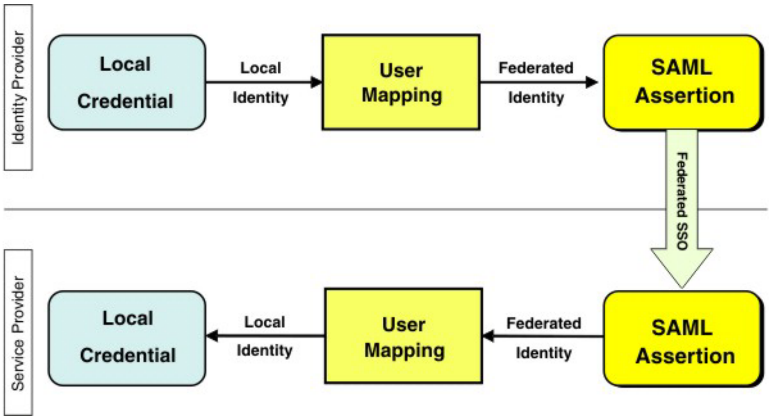


Figure 7. Example of identity mapping

Several methods can be used during the user mapping activity to achieve the required output identity. For example, hard-coded information can be added to the outgoing token, or Java code can be developed and used to acquire information from external sources and that information can be added to the outgoing token. This flexibility is achieved through *identity mapping rules* that are defined in either of two ways:

- An eXtensible Stylesheet Language Transformation (XSLT) file and processed by the Tivoli Federated Identity Manager Identity Mapping module.
- A custom mapping module that you create using Java.

Before you decide which method to use, you must understand how user identities are represented in Tivoli Federated Identity Manager, how tokens are processed, and how identities are mapped between partners.

Security Token Service (STS) Universal User document

In order to ensure that an incoming token can be converted properly into an outgoing token that contains the content and format that is required by the partner, Tivoli Federated Identity Manager creates an intermediate document in a generic XML format that holds identity information. This document is called the STS Universal User or STSUU. The STSUU document contains three sections:

- Principal information
- Group information
- Attribute information

To create the STSUU document, Tivoli Federated Identity Manager uses an XML schema that specifies the structure. The schema is defined in the file stsuuser.xsd. The following code sample contains the entire contents of the secure token service universal user XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:ibm:names:ITFIM:1.0:stsuuser"
xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser"
elementFormDefault="qualified">

 <xsd:element name="STSUniversalUser">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="Principal" type="stsuuser:PrincipalType"
             minOccurs="1" maxOccurs="1"/>
     <xsd:element name="GroupList" type="stsuuser:GroupListType"
             minOccurs="0" maxOccurs="1"/>
     <xsd:element name="AttributeList" type="stsuuser:AttributeListType"
             minOccurs="0" maxOccurs="1"/>
     <xsd:element name="RequestSecurityToken" type="stsuuser:RequestSecurityTokenType"
             minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" use="required"/>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="PrincipalType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="RequestSecurityTokenType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="AttributeType">
                  <xsd:sequence>
                   <xsd:element name="Value" type="xsd:string"
                         minOccurs="0" maxOccurs="unbounded"/>
                   </xsd:sequence>
                  <xsd:attribute name="name" type="xsd:string" use="required"/>
                  <xsd:attribute name="type" type="xsd:string" use="optional" />
                  <xsd:attribute name="nickname" type="xsd:string" use="optional" />
                   <xsd:attribute name="preferEncryption" type="xsd:boolean" use="optional" />
 </xsd:complexType>

 <xsd:complexType name="AttributeListType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="GroupListType">
  <xsd:sequence>
   <xsd:element name="Group" type="stsuuser:GroupType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="GroupType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="xsd:string" use="optional" />
 </xsd:complexType>

</xsd:schema>
```

Figure 8. STS Universal User document schema

Although the schema is used as the base for all STSUU documents, the exact information contained in any specific STSUU document is dependent on the token type for the security token that was used as input. The information required in an STSUU document after transformation by identity mapping depends on:

- The token type to be generated.
- The specific mapping rule being used for the conversion.

During token processing for a typical single sign-on configuration, two STSUUs are created. One is an input STSUU, which is created from the original input token. The other is an output STSUU, which is created after the identity mapping rules are applied. For more information, refer to “Token processing.”

Token processing

In a typical single sign-on configuration, tokens are processed by the Tivoli Federated Identity Manager trust service and three specific types of modules. When used in combination, the modules are referred to as a *trust chain*. Figure 9 provides a diagram of token processing. The input to the trust chain is the input security token. This token is created using the local credentials that are received when a user logs in. The first module in the trust chain converts the input token to an STSUU document. All attributes that are in the input token are available in the STSUU document. The STSUU document is now used as input to the identity mapping module. This module could be the Tivoli Federated Identity Manager mapping module that is used with an XSLT file or it could be a custom mapping module that you have created. A given mapping module may be used in common for many partners in the federation or one that is unique to a specific partner. The output of the mapping module is another STSUU document. This "output" STSUU document is used as input to the third token module, which converts the "output" STSUU to the output token. The output token is then sent to the partner.

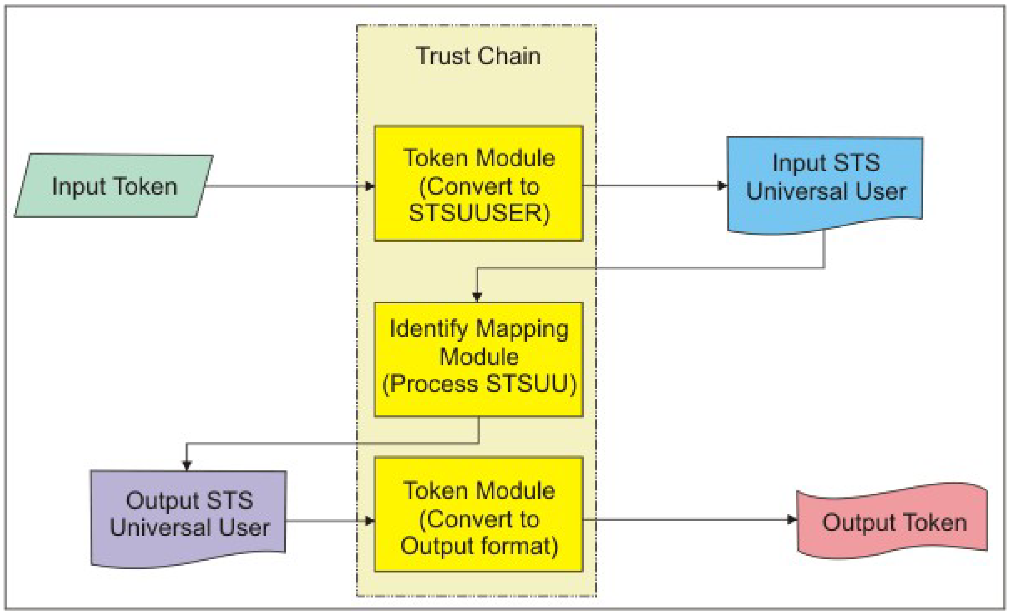

Trust Chain
Input Token
Token Module (Convert to STSUUSER)
Input STS Universal User
Identify Mapping Module (Process STSUU)
Output STS Universal User
Token Module (Convert to Output format)
Output Token

Figure 9. Token processing

Use of XSL language for creating mapping rules files

The identity mapping module uses the Java API for XML Parsing (JAXP) to transform the input document based on XSL configuration that you specify in an XSL file.

XSL is a language that can be used to transform (format) documents. XSL is used to define stylesheets for HTML and to format XML data so that it can be displayed in a Web browser. Part of the XSL standard defines transformations for moving data from one form to another. The transformation language can include conditional statements, variables, and call-outs to Java programs.

The trust service uses XSL as a language to create mapping rules that specify how to transform an input STS universal user document into an output STS universal user document that can be used to generate an output token. The XSL parser processes XSL documents by looking for matching templates. When a template is found, the contents of the template are processed.

The main tasks that are performed in mapping rules are:

- Move identity information between elements
- Reformat existing identity information
- Add new elements with new identity information
- Remove unwanted identity information

You can use the IBM Rational® Application Developer tool set to run an XSL debugger from a command line. This tool enables you to test your XSL code without having to run the trust service.

Tivoli Federated Identity Manager provides two sets of sample identity mapping files. The first set shows the minimum contents of each type of mapping. The location of these mapping files is:

```
/opt/IBM/FIM/examples/mapping_rules/
```

Table 16 lists the example mapping rules files:

Table 16. Example mapping rules

File name	Mapping description
ip_liberty.xsl	Tivoli Access Manager credential to Liberty token
ip_saml_1x.xsl	Maps a Local user identity to SAML 1.0 or SAML 1.1 token
ip_saml_20.xsl	Maps a local user identity to SAML 2.0 token using a token
ip_saml_20_email_nameid.xsl	Maps a local user identity to a SAML 2.0 token using the user's email address for the identity without an alias.
ip_wsfederation.xsl	Tivoli Access Manager credential to SAML token
ip_infocard.xsl	Incoming token to SAML 1.1 token
ip_openid.xsl	IVCred token to a Security Token Service Universal User (STSUU) token
rp_infocard.xsl	SAML 1.1 token to an IVCred token
sp_liberty.xsl	Liberty token to Tivoli Access Manager credential
sp_saml_20.xsl	Maps a SAML 2.0 token to a local user identity
sp_saml_1x.xsl	Maps a SAML 1.0 or 1.1 token to a local user identity

Table 16. Example mapping rules (continued)

File name	Mapping description
sp_saml_1x_ext.xsl	Maps a SAML 1.0 or 1.1 token to a local user identity and verifies that the authentication method is an acceptable one. It demonstrates that the service provider can require that the authentication used at identity provider be at a certain level. In this mapping rule, password authentication is not accepted. It throws an exception if password authentication was used.
sp_wsfederation.xsl	SAML token to Tivoli Access Manager credential
sp_tagvalue.xsl	SAML token to Tivoli Access Manager IV Cred credential with WebSEAL tag/value attributes
username_ivcred.xsl	Username token to Tivoli Access Manager credential

Note: For more information on the sample mapping rules for each protocol, see the protocol-specific configuration instructions in this guide.

The demonstration application provides sample XSL identity mapping rules files. These files expand upon the minimal mapping rules described above to perform mapping that is customized for the user accounts that are created by the demonstration application configuration scripts.

The location of the sample mapping scripts for the demonstration application is: `/opt/IBM/FIM/examples/demo/scripts/demo_rules/`

Note: The file names are the same as the minimal mapping rules, but the files are located in a different directory.

The sample mapping files are automatically installed during installation.

Table 17 lists the files for each federation type on each provider type.

Table 17. Sample mapping rules files for the demonstration application

Provider	Federation Type	Mapping rule file
Identity Provider	Liberty	`ip_liberty.xsl`
	SAML 1.0	`ip_saml_10.xsl`
	SAML 1.1	`ip_saml_11.xsl`
	SAML 2.0	`ip_saml_20.xsl`
	WS-Federation	`ip_wsfederation.xsl`
	Information Card	`ip_openid.xsl`
	OpenID	`ip_infocard.xsl`
Service Provider	Liberty	`sp_liberty.xsl`
	SAML 1.0 or 1.1	`sp_saml_1x.xsl`
	SAML 2.0	`sp_saml_20.xsl`
	WS-Federation	`sp_wsfederation.xsl`
	Information Card	`rp_infocard.xsl`
	OpenID	`sp_openid.xsl`

Tivoli Directory Integrator identity mapping module

This module performs generic user and attribute mapping functions. An assembly line executing on a Tivoli Directory Integrator (TDI) server is called to perform mapping of user and attribute data in an STSUniversalUser. Data may be resolved from a variety of data sources natively supported by TDI, including LDAP and relational databases. Custom code is also supported through JavaScript™ connectors.

Tivoli Federated Identity Manager provides a demonstration Tivoli Directory Integrator mappings file. The file is located with the other example files. For example, on Linux or UNIX, the file location is

```
/opt/IBM/FIM/examples/tdi_mappings/tdi_demo_mappings.xml
```

Deployment of this module requires:

- Configuration of the TDI trust module settings
- Configuration of the TDI server
- Configuration of SSL communication between the TDI server and the client (TDI trust module)

Complete the configuration instructions in:

- "Configuring the TDI trust module"
- "Configuring the TDI server" on page 128
- "Configuring SSL for Tivoli Directory Integrator trust module" on page 129

Configuring the TDI trust module

When you select the TDI security token module during creation a trust chain, you will be prompted to supply a number of configuration properties. The properties are described in this topic, and a worksheet is provided that you can consult when you use the administration console to configure your module.

Configuration properties

Server Hostname
Host name or IP address of the computer on which the Tivoli Directory Integrator server is running. The default value is localhost. For example, `tdiserver.company.com`.

Server Port
Port number on which the Tivoli Directory Integrator server is configured to run. The default value is 1099.

Assembly Line Handler Pool Size
Number of assembly line handlers to maintain for this trust chain. The value must be a positive integer. The default value is 10.

Number of Wait Threads
Maximum number of threads that can be waiting for an assembly line handler for this chain. The value must be a positive integer. The default value is 0.

Amount of time for threads to wait for an assembly line handler to become available
Determine the amount of time for threads to wait for an assembly line handler to become available. Select one of these options.

Wait indefinitely

Do not put a limit on the time for threads to wait for the assembly line handler to become available. This is the default choice.

Do not wait for assembly line handler after initial try

Do not allow any threads to wait for an assembly line handler and , if one is not available immediately, the Tivoli Directory Integrator module returns a timeout.

Use the following maximum wait value

Specify a value for the maximum time to wait.

Maximum Wait Time (milliseconds)

Maximum time a thread will wait for an assembly line handler before returning a wait timeout. This value is specified in milliseconds and it must be a positive integer.

Discover configuration settings

Use the server host name and port that were supplied earlier in this panel to connect to the Tivoli Directory Integrator server and discover which configurations and assembly lines are available. You must enter the Server Hostname and Server Port before you can select this option. After you select this option, two drop-down list boxes are available.

Select Configuration File

Select which configuration file to use from the list.

Select Assembly Line

Select which assembly line to use from the list. This list was derived from the configuration file you selected above.

Enter configuration settings manually

Enter the configuration settings manually by supplying the following fields:

Configuration File

Solution name, or the file name of the configuration file, to use. For example, `tdi_demo_mappings.xml`.

Assembly Line Name

Name of the assembly line to use. For example, `assemblyLine1`.

Table 18. Tivoli Directory Integrator Module configuration properties worksheet

Property	Your value
Server hostname	
Server Port	
Assembly Line Handler Pool Size	
Number of Wait Threads	
Amount of time for threads to wait for an assembly line handles to become available	Configuration panel provides three options: • Wait indefinitely • Do not wait for assembly line handler after initial try • Use the following maximum wait value: Maximum Wait Time (milliseconds)
Method for selecting the assembly line settings	Two choices: • Discover configuration settings • Enter configuration settings manually

Table 18. Tivoli Directory Integrator Module configuration properties worksheet (continued)

Property	Your value
Configuration file	
Assembly Line Name	

Configuring the TDI server

This topic describes the minimum procedure required for configuring a default installation of the TDI 6.1.1 server to be ready to run assembly lines for use with Tivoli Federated Identity Manager and the TDI STS module. The tdi_demo_mappings.xml is used as an example configuration.

For more detailed TDI configuration instructions, see the Tivoli Directory Integrator Information Center:http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_6.1.1/welcome.htm

The TDI installation prompts you to specify where TDI look for its solutions directory. The choices are:

- Use a TDI subdirectory under my home directory (default)
- Use Install Directory
- Select a directory to use

This procedure assumes the default of a TDI subdirectory under the home directory.

1. Establish solution files

   After initial installation, there will be a subdirectory under the home directory of the root user : /root/TDI. To populate solution files in this directory, start the TDI server without any parameters: once (or), and the solutions files will be populated:

   ```
   # /opt/IBM/TDI/V6.1.1/ibmdisrv
   ```

   As an alternative, you can start the TDI configuration editor. After the server starts, there should be files, including solutions.properties, in the /root/TDI directory
2. Update solution.properties.

   Make the following updates to /root/TDI/solutions.properties:

   api.remote.on
   This property allows use of the remote server API used by the TDI STS Module. Change the default value of false to true.

   api.remote.ssl.on
   These instructions show configuration of the TDI without SSL. Configuration of SSL is discussed later. Change the default value of true to false.

   api.remote.nonssl.hosts
   This property is needed when the TDI server is running on a different host from the Tivoli Federated Identity Manager runtime, and you are not using SSL between the Tivoli Federated Identity Manager runtime and the Tivoli Directory Integrator server. Specify the IP address of machine running the runtime (trust server).
3. Establish and populate the directory for TDI Configurations.

The solution.properties file contains a setting which describes the location for TDI configuration files that can be edited through the server API. This property, and its default value, is:

```
api.config.folder=/opt/IBM/TDI/V6.1.1/configs
```

You can choose a different directory if you want, but you must ensure that the directory is created, and any configuration file that you want TDI STS Module to use must be located in that directory. For example, you could create the directory, and copy the sample TDI configuration file into that directory, as follows:

```
# mkdir /opt/IBM/TDI/V6.1.1/configs
# cp /opt/IBM/FIM/examples/tdi_mappings/tdi_demo_mappings.xml
    /opt/IBM/TDI/V6.1.1/configs
```

4. Start the TDI server in daemon mode.

   Enter this command to start the server without SSL support:

   ```
   # /opt/IBM/TDI/V6.1.1/ibmdisrv -d
   ```

   The server should be running, and log information can be viewed in /root/TDI/logs/ibmdi.log.

The TDI STS Module is now able to load and run assembly lines.

Configuring SSL for Tivoli Directory Integrator trust module

The TDI STS module acts as a client to the TDI server. Configuration of SSL communication between the two can be done in a number of different ways. Configuration of the server and the client is done separately.

Security in general, and SSL configuration in particular is covered in great detail in the TDI documentation in the Tivoli Directory Integrator information center. There are several server API authentication scenarios available, but this document describes only mutually authenticated SSL. This is the recommended and supported deployment pattern for secured installations with the TDI STS Module.

TDI Server SSL configuration

Complete information can be found Tivoli Directory Integrator Administration Guide, under the topic TDI Server Instance Security.

The TDI Server needs two pieces of information for a mutually authenticated SSL configuration.

- A private key and certificate for the server's identity
- The public certificate or trusted signer of the client

The private key and certificate should be stored in a Java keystore file (JKS). For our example, server_identity.jks. The certificate alias of the private key in that keystore will be called tdi_server.

In this example, the JKS files have been created with the IBM IKeyman utility, which requires a keystore password, but does not create a separate key password for the individual private key. This is important when creating the TDI server stash file. In this example, the keystore password is `passw0rd`, and the private key password does not exist.

The public certificate/signer of the client should also be stored in a Java keystore. For our example, client_signer.jks. The certificate alias of the trusted signer certificate is not important for this configuration. The password used for the keystore is important, and this example uses `passw0rd`.

The distinguished name (DN) of the client certificate is important. In this example, the DN of the client cert is:

```
CN=tdi_client, O=ibm, C=US
```

To enable mutually authenticated SSL support on the TDI server, you must:

- Update properties in solution.properties
- Create the TDI Server stash file
- Update the TDI Server registry to recognize the DN of the client as an administrator
- Start the TDI server and validate SSL is in use

1. Modify solution.properties for TDI Server SSL Support

   Update the following parameters in solution.properties:

   com.ibm.di.server.keystore
   Specifies the keystore containing the private key and certificate of the TDI Server. The server_identity.jks file should be located in the solutions directory (/root/TDI for our example). Change the default of testserver.jks to server_identity.jks

   com.ibm.di.server.key.alias
   This is the certificate alias in the server keystore file which represents the private key of the server. Change the default of server to tdi_server.

   api.truststore
   Specifies a keystore containing trusted signer certificates and CA for server API clients. The client_signer.jks file should be located in the solutions directory (/root/TDI for our example). Change the default of testserver.jks to client_signer.jks.

   {protect}-api.truststore.pass
   This is the keystore password for the keystore specified in the api.truststore property. By prefixing with {protect}-, it will be automatically encrypted the next time the server is run. Change the default of {encr}-*key_string* to `passw0rd`.

   api.remote.ssl.on
   Set this to true to enable SSL.

2. Create the TDI Server stash file.

   The TDI server stash file is idisrv.sth. This file is located in the solutions directory. This file can contain one or two passwords. The first opens the keystore containing the server identity (server_identity.jks) and the second (optional) password is the password for the key itself within that keystore. If not specified, the second password is assumed to be the same as the first.

   When using the IBM iKeyman utility to create a self-signed certificate in a keystore file, the keystore password is manually specified when you create the keystore. However there is *no* private key password for the private key. This means you must create the TDI server stash file with a keystore password, and specifically set the private key password to null (the empty string), as follows:

   ```
   # /opt/IBM/TDI/V6.1.1/bin/createstash.sh passw0rd ""
   ```

3. Update the TDI Server registry to recognize the DN of the client as an administrator.

The TDI Server performs authorization on authenticated server API requests through a user registry and its assigned roles. The default registry is a text file, located in:

```
<solutions_directory>/serverapi/registry.txt
```

Add the following text to registry.txt:

```
[USER]
[ID]:CN=tdi_client, O=ibm, C=US
[ROLE]:admin
[ENDUSER]
```

For more advanced registry configuration, see the Tivoli Directory Integrator information center.

4. Start the TDI Server and validate that SSL is on.

   Start the TDI server, and validate that the startup message indicates SSL is in use, as follows:

   ```
   # /opt/IBM/TDI/V6.1.1/ibmdisrv -d
   CTGDKD024I Remote API successfully started on port:1099,
   bound to:'SessionFactory'. SSL and Client Authentication
   are enabled.
   ```

Server side SSL configuration is complete.

Client-side SSL configuration

The TDI STS Module acts as an SSL client and can operate in one of two configurations:

- It can leverage the WebSphere Application Server JSSE configuration for SSL support.

  Note: You must use this option with embedded WebSphere.
- You can specify Java system properties which control which key store and trust store the TDI Server API uses.

  Note: This option does not work with embedded WebSphere.

For both options, the client needs two pieces of information for a mutually authenticated SSL configuration:

- A private key and certificate for the client identity.
- The public certificate or trusted signer of the server.

For this example, store the private key and certificate in a Java keystore file called client_identity.jks, and store the certificate alias of the private key in a keystore called tdi_client. In this example, the JKS file has been created with the IBM IKeyman utility and the keystore password is passw0rd. IKeyman does not assign a second password assigned to the key itself. It is necessary to assign a password for the key in order to successfully start the JVM. Use the same password as the keystore password. You can modify the keystore that was created with IKeyman using the Java keytool parameter to assign a password to the key, as follows:

```
# /opt/IBM/WebSphere/AppServer/java/bin/keytool -keypasswd
-alias tdi_client -new passw0rd -keystore client_identity.jks
-storepass passw0rd
```

The public certificate/signer of the server is also stored in a Java keystore. For our example, use server_signer.jks. The certificate alias of the trusted signer certificate is not important for this configuration, but the keystore password keystore is needed. Set the password to passw0rd.

We will use the Federated Identity Manager key service to import both keystore files into the Tivoli Federated Identity Manager portion of the WebSphere configuration repository, for future reference.

Use the Key Service menu of the administration console to import the client_identity.jks and server_signer.jks files.

On the GUI panel for the client_identity.jks, specify a keystore name of tdi_client, and a Type of Signing/Encryption keys.

On the GUI panel for server_signer.jks, specify a keystore name of tdi_server, and a Type of CA certificates.

The files actually appear in the WebSphere configuration repository file system at:

```
<config_root>/itfim/<fim_domain>/etc/jks/tdi_client.jks
<config_root>/itfim/<fim_domain>/etc/jks/tdi_server.jks
```

This configuration example uses a fim_domain value of `idp`.

The next step is use one of two methods for client-side SSL configuration of the TDI STS Module:

- Using the WebSphere JSSE configuration
- Using Java system properties to specify keys

You do not need to do both methods.

Client-side SSL using the WebSphere JSSE configuration

This topic summarizes information described in detail in the following locations:

- WebSphere information center. For example, for WebSphere Application Server, see the topic *Dynamic outbound selection of Secure Sockets Layer configurations*
- Developer works article: *SSL, certificate, and key management enhancements for even stronger security in WebSphere Application Server V6.1*:

  http://www-128.ibm.com/developerworks/websphere/techjournal/0612_birk/0612_birk.html?ca=drs-

We cannot use the dynamic outbound endpoint SSL configurations as this requires the SSL client to utilize the WebSphere JSSEHelper class to set specific connection information parameters, and TDI uses only standard Java JSSE interfaces.

Consequently we must modify the scoped SSL configuration for the server which is running the Tivoli Federated Identity Manager Runtime. Depending on whether you are running a cluster or a standalone application server, you could make this change at either cell or node level. Our example uses a standalone application server, and modifies the node default key store and node default trust store.

For our deployment pattern we will:

- Update the NodeDefaultKeyStore by importing the client private key and certificate
- Update the NodeDefaultTrustStore by importing the server's public certificate

To import the client private key and certificate into the NodeDefaultKeyStore, use the WebSphere administration console:

1. Navigate to **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates**.
2. Press Import to import a new key and enter values:

   Key file name
   /opt/IBM/WebSphere/AppServer/profiles/idp/config/itfim/idp/etc/jks/tdi_client.jks

   Type
   JKS

   Key file password
   passw0rd

   Now press **Get key file aliases**

   Certificate alias to import
   tdi_client

   Imported certificate alias
   tdi_client
3. After the import, the key should be displayed in the Alias column as tdi_client. Save the WebSphere configuration after loading the key.

Before we can import the server's public certificate into the NodeDefaultTrustStore, we need the server certificate in a simple file format (PEM ascii format or DER binary format) rather than in the JKS. Use IKeyman, or keytool, to export the server's public certificate from the file: <*config_root*>/itfim/<*fim_domain*>/etc/jks/tdi_server.jks

For example, you can export the public key using IKeyman into a PEM ascii format in a file called: /root/keys/tdi_server.arm.

To import the server public certificate into the NodeDefaultTrustStore, use the WebSphere Administration Console as follows:
1. Navigate to **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates**.
2. Press **Add** to add a certificate.

   The Add Signer Certificate panel is displayed.

   Alias
   tdi_server

   File name
   /root/keys/tdi_server.arm

   Data type
   Base-64 encoded ASCII data
3. The certificate named tdi_server should now be in the list of certificates. Save the WebSphere configuration after committing this change.
4. Restart WebSphere, and the client should be configured for SSL.

Client-side SSL using Java system properties

The use of the Java system properties for client-side SSL is described in the Tivoli Directory Integrator Information Center. In the TDI Administrator Guide, see the topic *Server API Access Security*

Note: This option is not available for embedded WebSphere installations.

These Java system properties can be used select key stores and certificates for SSL communications:

api.client.ssl.custom.properties.on
Instructs the TDI server API to use these custom properties for keystore/truststore configuration rather than the JSSE configuration. Example value: true

api.client.keystore
Specifies the keystore containing the client certificate. Example value:

```
${USER_INSTALL_ROOT}/config/itfim/idp/etc/jks/tdi_client.jks
```

api.client.keystore.pass
Password for the file specified in api.client.keystore. Example: passw0rd

api.client.key.pass
Password for the actual key in api.client.keystore.

Leave unspecified as we used the keytool utility to make the key password the same as the keystore password.

api.truststore
Specifies the keystore containing the TDI Server public certificate. Example:

```
${USER_INSTALL_ROOT}/config/itfim/idp/etc/jks/tdi_server.jks
```

api.truststore.pass
Password for the file specified in api.truststore. Example: passw0rd

Using the WebSphere Administration Console, update the server's JVM startup parameters:

1. Navigate to **Servers > Application Servers > server1 > Java and Process Management > Process Definition > Java Virtual Machine.**
2. Update the properties:

   Generic JVM arguments:
   -Dapi.client.ssl.custom.properties
3. Restart WebSphere, and the client should be configured for SSL.

Creating a custom mapping module

Before you begin

Creating a custom mapping module is a programming-intensive procedure that involves writing a Java class and installing the class into the plug-ins directory for your domain. To create a custom mapping module, you must be familiar with the structure of Tivoli Federated Identity Manager trust service modules and the proper procedures for creating them and adding them to your environment.

About this task

Learn more about trust service modules in:

- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions* (SG24-6394-01). This book is available in PDF (Portable Document Format) at ../http://www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf or in HTML (Hypertext Markup Language) at ../http://www.redbooks.ibm.com/redbooks/SG246394/

- A developerWorks article titled *Tivoli Federated Identity Manager: Implementing and deploying custom trust modules* at http://www-128.ibm.com/developerworks/tivoli/library/t-sts-custom/

Adding a custom mapping module

To add a custom mapping module that you have created, you must first define the module as a new module type in the Tivoli Federated Identity Manager environment.

Before you begin

You must write a Java class for a new module type and install the class into the plug-ins directory for your domain. You can then use the following instructions to create a new module type entry in the console.

About this task

This task is necessary only when the XSL Transformation module that is supplied with Tivoli Federated Identity Manager does not meet the requirements of your deployment.

Procedure

1. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Runtime Node Management**. The Runtime Node Management panel is displayed.
2. Click the **Publish plug-ins** button.
3. When prompted, click the **Load configuration changes to Tivoli Federated Identity Manager runtime**. The new module type is displayed in the Module Type list.

What to do next

Continue with the task for adding an instance of the mapping file in "Adding an instance of a custom mapping module."

Adding an instance of a custom mapping module

After you have created your mapping module and added it as a module type, you must create an instance of that module type in order to use it in the Tivoli Federated Identity Manager environment.

Before you begin

Be sure that you have completed the following tasks before continuing with these instructions:

- "Creating a custom mapping module" on page 134
- "Adding a custom mapping module"

About this task

The console provides a wizard to guide you through adding the module instance.

Procedure

1. Click **Tivoli Federated Identity Manager** → **Configure Trust Service** → **Module Instances**. The Module Instances panel displays module instances that are created by default, and also displays any module instances that you have added.
2. Click **Create**. The Token Type panel displays the module types that have been defined. The list includes the default token types and any custom token types that you have defined.
3. Select a token type and click **Next**. The Module Instances wizard displays the Module Instances Name panel.
4. Enter values for the requested properties and click **Finish**. See the online help for descriptions of the fields.

What to do next

Now the new mapping file will be available in the list of modules that you can choose from when establishing a federation.

Chapter 13. SAML federations overview

Tivoli Federated Identity Manager supports the following OASIS Security specifications for exchanging information in a federation:

- SAML 1.0 and 1.1 (referred to as 1.x)
- SAML 2.0

SAML (Security Assertion Markup Language) is an XML standard for exchanging single sign-on information. It relies on the use of SOAP among other technologies to exchange XML messages over computer networks. The messages are exchanged through a series of requests and responses, in which one of the federation partners sends a request message to the other federation partner. Then, that receiving partner immediately sends a response message to the partner who sent the request.

The SAML specifications include the following descriptors to specify the structure and content of the messages and the way in which the messages are to be communicated between partners and end-users to establish a federation and initialize and manage single sign-on:

Assertions
XML-formatted tokens that are used to transfer user identity information, such as the authentication, attribute, and entitlement information, in the messages.

Protocols
The types of request messages and response messages that are used for obtaining authentication data and for managing identities.

Bindings
The communication method used to transport the messages.

Profiles
Combinations of protocols, assertions, and bindings that are used together to create a federation and enable federated single sign-on.

When using Tivoli Federated Identity Manager, you and your partner must use the same SAML specification (1.0, 1.1, or 2.0) and must agree on which protocols, bindings, and profiles to use.

The sections that follow give brief descriptions of how SAML 1.x and SAML 2.0 specifications are used in Tivoli Federated Identity Manager. However, these descriptions do not provide all of the details of the specifications. Refer to the OASIS specification documents at http://www.oasis-open.org/specs/index.php for those details.

SAML 1.x

Tivoli Federated Identity Manager supports both SAML 1.0 and SAML 1.1. These specifications are referred to collectively as SAML 1.x.

If you and your partner choose to use SAML 1.x in your federation, you will need to understand the SAML 1.x support that is provided in Tivoli Federated Identity Manager.

Assertions

The assertions created by Tivoli Federation Identity Manager contain authentication statements, which assert that the principal (that is, the entity requesting access) was authenticated. Assertions can also carry attributes about the user that the identity provider wants to make available to the service provider.

Assertions are usually passed from the identity provider to the service provider.

The content of the assertions created by Tivoli Federated Identity Manager is controlled by the specification (SAML 1.0 or 1.1) that you select when you establish a federation and by the definitions used in the Tivoli Federated Identity Manager identity mapping method (either a custom mapping module or an XSL transformation file) that you configure. The identity mapping also specifies how identities are mapped between federation partners.

Protocol

In Tivoli Federated Identity Manager, SAML 1.x uses a simple request-response protocol to make authentication requests.

Binding

SAML 1.x both plain HTTP (using browser redirects) or SOAP for the transportation of messages. The *profile* used in the federation further specifies how the communication of the messages takes place.

Profiles

SAML 1.x specifies two options for profiles:

Browser artifact

Browser artifact uses SOAP-based communications (also called the SOAP backchannel) to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

Browser POST

Browser POST uses a self-posting form during the establishment and use of the trusted session between an identity provider, a service provider, and a client (browser).

Tivoli Federated Identity Manager supports browser artifact by default when you select SAML 1.0 or SAML 1.1 as the profile for your federation. However, you can use browser POST in your federation on a per-partner basis. For example, if you are a service provider, you can specify that your identity provider partner uses Browser POST when you configure that partner. If you are an identity provider, you can enable the IBM PROTOCOL extension when configuring a SAML 1.x federation.

The URL that is used to initiate single sign-on differs depending on whether the identity provider is using this extension. For more information about URLs, see "SAML 1.x initial URL" on page 499.

SAML 2.0

The SAML 2.0 specification introduced more flexibility than the previous SAML 1.x specifications.

Assertions

The assertions created byTivoli Federated Identity Manager contain authentication statements. These authentication statements assert that the principal (that is, the entity requesting access) was authenticated. Assertions can also carry attributes about the user that the identity provider wants to make available to the service provider.

Assertions are typically passed from the identity provider to the service provider.

The content of the assertions that are created is controlled by the specification (SAML 2.0). You select these assertions when you establish a federation. You also select these assertions by the definitions used in the Tivoli Federated Identity Manager identity mapping method that you configure. The identity mapping method can either be a custom mapping module or an XSL transform file. The identity mapping also specifies how identities are mapped between federation partners.

Protocols

SAML 2.0 defines several request-response protocols, all correspond to the action being communicated in the message. The SAML 2.0 protocols that are supported in Tivoli Federated Identity Manager are:

- Authentication request
- Single logout
- Artifact resolution
- Name identifier management

Bindings

When you use SAML 2.0 in Tivoli Federated Identity Manager, you have several binding options. These options specify the way in which messages can be transported:

HTTP redirect

HTTP redirect enables SAML protocol messages to be transmitted within URL parameters. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary. The intermediary might be necessary if the communicating entities do not have a direct path of communication. The intermediary might also be necessary if the responder requires interaction with a user agent such as an authentication agent.

HTTP redirect is sometimes called browser redirect in single sign-on operations. This profile is selected by default.

HTTP POST

HTTP POST enables SAML protocol messages to be transmitted within an HTML form using base64-encoded content. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary. The agent might be necessary if the communicating entities do not have a direct path of communication. The intermediary might also be necessary if the responder requires interaction with a user agent such as an authentication agent.

HTTP POST is sometimes called Browser POST, particularly when used in single sign-on operations. It uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

HTTP artifact
HTTP artifact is a binding in which a SAML request or response (or both) is transmitted by reference using a unique identifier called an artifact. A separate binding, such as a SOAP binding, is used to exchange the artifact for the actual protocol message. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary. This setting is used when it is not preferable to expose the message content to the intermediary.

HTTP artifact is sometimes called browser artifact, particularly when used in single sign-on operations. The HTTP artifact uses a SOAP back channel. The SOAP back channel is used to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

SOAP SOAP is a binding that uses Simple Object Access Protocol (SOAP) for communication.

The choice of binding you have depends on the profile you choose to use in your federation.

Profiles

Tivoli Federated Identity Manager supports the configuration of the single sign-on profile on a per-partner basis. The profiles supported are:

Web browser single sign-on
The Web Browser SSO profile is the consolidation of the browser artifact and browser POST profiles that were introduced in SAML 1.x. Using this profile, an authentication request message is sent from a service provider to an identity provider. A response message containing a SAML assertion is sent from the identity provider to the service provider. Additional messages are sent related to artifact resolution, if that binding is used. This profile provides options regarding the initiation of the message flow and the transport of the messages:

Message initiation
The message flow can be initiated from the identity provider or the service provider.

Bindings
In a Tivoli Federated Identity Manager environment, the following bindings can be used in the Web browser SSO profile:

- HTTP Redirect (available only in an identity provider configuration)
- HTTP POST
- HTTP artifact

The choice of binding depends on the type of messages being sent. For example, an authentication request message can be sent from a service provider to an identity provider. The response message can be sent from an identity provider to a service provider using either HTTP POST or HTTP artifact. A pair of partners in a federation do not need to use the same binding.

Options

The Web Browser single sign-on profile in Tivoli Federated Identity Manager also provides the following option:

Enhanced Client Proxy This profile option enables an enhanced client or proxy (ECP) to communicate with an identity provider and service provider on behalf of a user (client). For example, a user might request a resource from a service provider. The service provider might not know which identity provider to access in order to authenticate the user. Using the ECP profile option, the service provider can contact the ECP, which knows how to locate and access the appropriate identity provider. The ECP profile supports SOAP and reverse SOAP (PAOS) bindings during the processing of authentication requests.

Single Logout

The Single Logout profile is used to terminate all the login sessions currently active for a specified user within the federation. A user who achieves single sign-on to a federation establishes sessions with more than one participant in the federation. The sessions are managed by a session authority, which in many cases is an identity provider. When the user wants to end sessions with all session participants, the session authority can use the single logout profile to globally terminate all active sessions.

Message initiation

The message flow can be initiated from the identity provider or the service provider.

Bindings

In a Tivoli Federated Identity Manager environment, the following bindings can be used in the Single Logout profile:

- HTTP Redirect
- HTTP POST
- HTTP artifact
- SOAP

Name Identifier Management

The Name Identifier Management profile manages user identities that are exchanged between identity providers and service providers. The profile enables identity providers to notify service providers. Service providers are notified when there is a change to the content or format of an identity for a given user (principal). The profile enables service providers to specify unique *aliases* for the principal. Service providers can also send those aliases to the identity provider to be used instead of the principal name. The profile also enables either provider. The profile notifies its partner when it decides to no longer issue or accept messages that use the identity of the principal.

To manage the aliases, Tivoli Federated Identity Manager uses a function called the *alias service*. The alias service stores and retrieves aliases that are related to a federated identity. Aliases can be used in the following ways:

Persistent aliases

When persistent aliases are used, the identity of the user is federated by the identity provider to the identity of the user at the service provider. A persistent SAML name identifier is used. The user remains in the federation permanently, that is, until a request is made to terminate the federation.

Transient aliases
When transient aliases are used, a temporary identifier is used to federate between the identity provider and service provider. A temporary identifier is used only for the life of the user's single sign-on session.

In a Tivoli Federated Identity Manager environment, aliases are stored in and retrieved from one of the following types of repositories:

- An LDAP database.
- A relational database that supports JDBC.

During the configuration of Tivoli Federated Identity Manager, you can configure your environment to use one of these repository types.

Message initiation
The message flow can be initiated from the identity provider or the service provider.

Bindings
The following bindings can be used in the Name Identifier Management profile:

- HTTP Redirect
- HTTP POST
- HTTP artifact
- SOAP

Identity Provider Discovery
The Identity Provider Discovery profile is used by service providers to discover which identity provider is used by a user (principal) during Web browser single sign-on. Some deployments have more than one identity provider, and the service provider must be able to determine which identity provider a principal uses. The Identity Provider Discovery profile uses a cookie. The cookie is created in a domain that is common between identity providers and service providers in a given deployment. The cookie contains the list of identity providers and is called the *common domain cookie*.

When you configure your federation using the Tivoli Federated Identity Manager console, your profile options are:

Basic: Web Browser SSO, Single Logout
This setting enables the following profiles and bindings:

- Web Browser single sign-on with HTTP POST and HTTP Artifact bindings.
- Single logout, with HTTP POST and HTTP Artifact bindings.

Typical: Web Browser SSO, Single Logout, and Name Identifier
This setting enables the following profiles and bindings:

- Web Browser single sign-on, with HTTP POST and HTTP Artifact bindings.
- Single logout, with HTTP POST and HTTP Artifact bindings.
- Enhanced client or proxy

- Name Identifier Management, with HTTP POST and HTTP Artifact bindings.

Enable all profiles and bindings
This setting enables all the available profiles and bindings:
- Web Browser single sign-on, with HTTP POST, HTTP Artifact, and HTTP Redirect bindings.

  Note: HTTP Redirect is available only in an identity provider configuration.
- Enhanced client or proxy.
- Single logout, with HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- Name Identifier Management, with HTTP Redirect, HTTP POST, HTTP Artifact, and SOAP
- Identity Provider Discovery

Manual: Choose individual profiles and bindings
All supported profiles and available bindings are presented so that you can choose the ones you want to use.

Account linkage

In SAML 2.0, account linkage enables a user to link an identity provider account to a service provider. The link happens during the single sign-on initiated at the identity provider and service provider. In both scenarios, account linkage requires a user to be authenticated at both the service provider and identity provider.

An administrator can enable this feature in the partner settings panel. If this feature is enabled, the user must authenticate in the service provider when a persistent alias is received. The alias can not have been previously linked to an account in the service provider for the authentication to occur.

After the user authenticates, the SAML 2.0 implementation stores the alias at the service provider alias service and establishes account linkage.

Handling an unknown alias

SAML 2.0 supports aliases to communicate user identities between partners.

An administrator can configure the SAML 2.0 partner settings to handle an unknown alias in one of the following ways:
- The authentication page displays an error page when the service provider does not know the alias received from the identity provider. This setting is the default when you
  - Do not select **Force authentication to achieve account linkage**.
  - Do not select **Map unknown name identifiers to the anonymous username**.
- The SAML 2.0 implementation maps the identity of the user to the default user account. A guest account establishes the single sign-on session. This setting requires that you
  - Do not select **Force authentication to achieve account linkage**.
  - Select **Map unknown name identifiers to the anonymous username**.
- The user must authenticate at the service provider, which enables account linkage. This setting requires that you

- Select **Force authentication to achieve account linkage**.
- Do not select **Map unknown name identifiers to the anonymous username**.

Chapter 14. SAML endpoints and URLs

Communications within a federation take place through endpoints on the servers of the identity provider and service provider partners. In a Tivoli Federated Identity Manager environment, endpoints fall into two categories:

- Endpoints that are specified by the federation specification (such as SAML 1.x or SAML 2.0) and are used for partner-to-partner communication.
- Endpoints that end users can access to initiate a single sign-on activity.

All endpoints can be accessed through URLs. The syntax of the URLs is specific to the purpose of the access and whether the access is by a partner or by an end user.

URLs for partner communication

The URLs that are used for partner-to-partner communication, such as the exchange of requests, in both SAML 1.x and SAML 2.0 federations are referred to collectively as *endpoint URLs* or individually by the name of the protocol and binding or service that they are related to. Administrators who are responsible for installing, configuring, and maintaining the Tivoli Federated Identity Manager environment and the partner-to-partner communication in that environment will see references to these endpoint URLs and might find it helpful to understand their purpose. See “SAML 1.x endpoints and URLs” on page 146 or “SAML 2.0 endpoints and URLs” on page 148.

URLs for user access

While the SAML specifications define the endpoints for partner-to-partner communication, they provide limited or no guidance about the endpoints or methods that end users must use to initiate single sign-on actions. Tivoli Federated Identity Manager supports specific URLs for end-user initiation of single sign-on actions.

In a SAML 1.x federation, the single sign-on process is always initiated at the *intersite transfer service*. The method by which the request arrives at this endpoint is not specified in the SAML specification. The syntax for the intersite transfer service URL in a Tivoli Federated Identity Manager environment is described in “SAML 1.x initial URL” on page 499.

In a SAML 2.0 federation, single sign-on actions can be initiated at the identity provider site or the service provider site. URLs that can be used by users to initiate a sign-on action are specific to the a single sign-on action (such as initiate a federated sign on, perform a single logout, or end account linkage) and to whether the action is being initiated at the identity provider or service provider site. In a Tivoli Federated Identity Manager environment, the URLs that can be used for initiating sign-on actions are referred to as *profile initial URLs*. Architects and application developers, who will design and implement their users' interactions with the single sign-on process, will need to understand profile initial URLs. See “SAML 2.0 profile initial URLs” on page 501.

SAML 1.x endpoints and URLs

Several endpoints are configured on your point of contact server so that communications can be exchanged between you and your partner. These endpoints are configured when you configure your federation in Tivoli Federated Identity Manager. The endpoints are accessible through URLs and are used by the partners in the federation. If you are responsible for installing, configuring, or maintaining a federation in Tivoli Federated Identity Manager, you might find it helpful to be familiar with these endpoints and URLs.

The following endpoints are used in a SAML 1.x federation.

Point of contact server

The endpoint on the point of contact server where communication takes place. The syntax of the point of contact server URL is:

```
https://hostname:port_number
```

Where:

https https might be http if SSL is not enabled on the server.

hostname

The hostname of the point of contact server.

port_number

The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

You will be prompted for your point of contact server URL when you configure your federation. After the configuration, your point of contact server URL has /sps appended to it so that the syntax of the configured point of contact server URL is

```
https://hostname:port_number/sps
```

The /sps indicates that the URL is defined for single sign-on services.

Intersite transfer service

The endpoint on the identity provider point of contact server where the sign-on request process begins. This is the location to which users' single sign-on requests are sent. SAML does not specify how the requests arrive at this endpoint. If you are an identity provider using Tivoli Federated Identity Manager, the method used depends on how and where the users are logging in. For example, if the users will log in at the service provider partner Web site, your service provider partner will need the URL for your intersite transfer service and will then need to configure some type of redirect that takes the users from their site to your login form.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/saml1xx/login
```

Where:

https https might be http if SSL is not enabled on the server.

hostname

The hostname of the point of contact server.

port_number

The port number where communications take place on the server.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml*xx*
The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login The designation of what type of endpoint is using the port. **login** is used for the intersite transfer service in SAML 1.x federations.

This endpoint is used only on identity provider configurations and is defined automatically for you when you configure your federation.

Artifact resolution service
The endpoint on the identity provider point of contact server where artifacts are exchanged for assertions. This endpoint is the location where the federation partners communicate. It is sometimes referred to as the *SOAP endpoint* on the identity provider's point of contact server.

Note: You might also be familiar with this endpoint as the *responder service*.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/samlxx/soap
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number is 9444.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml*xx*
The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

soap The designation of what type of endpoint is using the port. **soap** is used for the artifact resolution service in SAML 1.x federations.

This endpoint is used only on identity provider configurations and is defined automatically for you when you configure your federation.

Assertion consumer service
The endpoint on the service provider point of contact server that receives assertions or artifacts. This endpoint is the location where the federation

partners communicate. This endpoint is sometimes referred to as the *SOAP endpoint* on the service provider's point of contact server.

Note: If you are using the browser artifact profile, you might be familiar with this endpoint as the *artifact consumer service* or the *artifact receiver service*.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/samlxx/login
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml*xx*
The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service.

This endpoint is used only on service provider configurations in SAML 1.x federations and is defined automatically for you when you configure your federation.

SAML 2.0 endpoints and URLs

Several endpoints are configured on your point of contact server so that communications can be exchanged between you and your partner. These endpoints are configured when you configure your federation in Tivoli Federated Identity Manager. The endpoints are accessible through URLs and are used by the partners in the federation. If you are responsible for installing, configuring, or maintaining a federation in Tivoli Federated Identity Manager, you might find it helpful to be familiar with these endpoints and URLs.

The following endpoints are used in a SAML 2.0 federation.

Point of contact server
The endpoint on the point of contact server where communication takes place. The point of contact server URL is also used as the provider ID. The syntax of the point of contact server URL is:

```
https://hostname:port_number
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

You will be prompted for your point of contact server URL when you configure your federation. After the configuration, your point of contact server URL has /sps appended to it so that the syntax of the configured point of contact server URL is

```
https://hostname:port_number/sps
```

The /sps indicates that URL is defined for single sign-on services.

Artifact resolution service (or SOAP endpoint)
The endpoint on either the identity provider or service provider point of contact server where artifacts are exchanged for SAML messages. This endpoint is the location where the federation partners communicate. It is sometimes referred to as the *SOAP endpoint*.

Note: You might also be familiar with this endpoint as the *responder service*.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/saml20/soap
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number is 9444.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

soap The designation of what type of endpoint is using the port. **soap** is used for the artifact resolution service in SAML 2.0 federations.

This endpoint is defined automatically for you when you configure your federation.

Assertion consumer service
The endpoint on the service provider point of contact server that receives assertions or artifacts. This endpoint is the location where the federation partners communicate.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/saml20/login
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

login The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service in SAML 2.0 federations.

This endpoint is used only on service provider configurations in SAML 2.0 federations and is defined automatically for you when you configure your federation.

Single sign-on service
The endpoint on the identity provider point of contact server that receives authentication requests.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/saml20/login
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

login The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service in SAML 2.0 federations.

This endpoint is used only on identity provider configurations in SAML 2.0 federations and is defined automatically for you when you configure your federation.

Single logout service

The endpoint on the identity provider or service provider point of contact server that receives logout requests.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

```
https://hostname:port_number/sps/federation_name/saml20/slo
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

The port is assigned with the default value unless the port is unavailable when Tivoli Federated Identity Manager is installed. If the default port is unavailable, the installation program adds a value of 1 to the port number until it finds an available port of that number.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

slo The designation of what type of endpoint is using the port. **slo** is used for the single logout service in SAML 2.0 federations.

Name identifier management service

The endpoint on the identity provider or service provider point of contact server that receives messages related to name management. The URL is based on the URL that you specify for the point of contact server and on the binding that is used.

The syntax for HTTP Redirect, HTTP POST, and HTTP Artifact is:

```
https://hostname:port_number/sps/federation_name/saml20/mnids
```

The syntax for SOAP is:

```
https://hostname:port_number/sps/federation_name/saml20/soap
```

Where:

https https might be http if SSL is not enabled on the server.

hostname
The hostname of the point of contact server.

port_number
The port number where communications take place on the server. The port depends on the binding being used. The default ports are:

HTTP SOAP: 9444

HTTP POST, HTTP Artifact, HTTP Redirect: 9443

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

mnids or soap
The designation of what type of endpoint is using the port. **mnids** is used for the name identifier management service in SAML 2.0 federations that use HTTP Redirect, HTTP POST, or HTTP Artifact. **soap** is used when SOAP is used as the binding.

Common Domain Cookie Service URL used by the Identity Provider Discovery service

By default, Tivoli Federated Identity Manager provides a common domain service implementation that makes it possible for an identity provider to let a service provider know that a specific user is ready to use a federation. The default URL is used internally and specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter (the identity provider) or cdcreader (the service provider) appended to the end of the URL. The default syntax for the URL is: :

```
https://common_domain_name/sps/federation_name/saml20/[cdcreader|cdcwriter}
```

Where:

https https might be http if SSL is not enabled on the server.

common_domain_name
The shared common domain name.

sps The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

cdcwriter *or* **cdcreader**
The designation of what type of action (read/get or write/set) is used.

Note: Tivoli Federated Identity Manager also supports the use of a third-party or custom discovery service.

Chapter 15. Sample identity mapping rules for SAML federations

The following topics show the sample identity mapping rules that are provided for SAML federations. If you have decided to use identity mapping rules for your federation, you can review the XSLT rules.

For an overview of identity mapping, including discussion of identity mapping options that do not use XSLT mapping rules files, see PlanningIdentityMapping.dita

- "Mapping a local user identity to a SAML 1.x token"
- "Mapping a SAML 1.x token to a local user identity" on page 154
- "Mapping a local identity to a SAML 2.0 token using an alias" on page 155
- "Mapping a SAML 2.0 token to a local identity" on page 156

Mapping a local user identity to a SAML 1.x token

This scenario occurs when messages are exchanged between partners in a SAML 1.0 or SAML 1.1 single sign-on federation. When a user request is received (for example, for access to a remote resource) Tivoli Federated Identity Manager contacts the point of contact server (for example, WebSphere Application Server) and obtains a local user identity.

The Tivoli Federated Identity Manager server places the local user identity information into an XML document that conforms to the security token service universal user (STSUUSER) schema. The server then consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

The identity mapping module then modifies the XML document to contain the information required to build a SAML token.

Table 19. STSUUSER entries used to generate a SAML token

STSUUSER element	SAML Token Information	Required?
Principal Attr: Name	AuthenticationStatement/Subject/NameIdentifier	Required
Attribute List	Additional custom attributes	Optional

The mapping module is responsible for two tasks:

1. Mapping Principal Attr Name to a Principal Name entry.

   The type must be valid for SAML. For example:

   ```
   urn:oasis:names:tc:SAML:1.0:assertion#emailAddress
   ```

   Figure 10 on page 154 shows part of the default mapping rule file, ip_saml_1x.xsl.

```
<!--
  This template replaces the entire Principal element with one that contains
  just the iv user name.
 -->
 <xsl:template match="//stsuuser:Principal">
  <stsuuser:Principal>
   <stsuuser:Attribute name="name" type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
    <stsuuser:Value>
     <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']
     [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:Principal>
 </xsl:template>
```

Figure 10. XSL code sample showing mapping of a local user identity into a Principal name for a SAML token

In this example, the local user identity is referred to as the *iv user name*.

```
<stsuuser:Value>
     <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']
     [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
```

2. Setting the authentication method to the `password` mechanism. This action is required by the SAML standard.

   See Figure 11.

```
<xsl:template match="//stsuuser:AttributeList">
     <stsuuser:AttributeList>
         <!-- First the authentcation method attribute -->
         <stsuuser:Attribute name="AuthenticationMethod"
              type="urn:oasis:names:tc:SAML:1.0:assertion">
            <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
         </stsuuser:Attribute>
     </stsuuser:AttributeList>
</xsl:template>
```

Figure 11. XSL code sample showing assignment of authentication method as an Attribute for a SAML token

Mapping a SAML 1.x token to a local user identity

The service provider receives a SAML 1.0 or SAML 1.1 token. Tivoli Federated Identity Manager converts the token contents into a XML file that conforms to the security token service universal user schema.

Table 20. SAML token information that is converted into a STS universal user document

SAML Token Information	STSUUSER element
AuthenticationStatement/Subject/NameIdentifier	Principal Attr: Name

Tivoli Federated Identity Manager converts this information to a local user identity.

- The NameIdentifier is used to populate the `name` attribute of the Principal.

  Figure 12 on page 155 shows the assignment of a set value for the Principal name. This code sample is from the default mapping file sp_saml_1x.xsl

```
<!--
  This will replace the principal name with the user's local name.
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
            <stsuuser:Value><xsl:value-of
select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value"/>
           </stsuuser:Value>
            </stsuuser:Attribute>
 </xsl:template>
```

Figure 12. XSL code sample showing assignment of a value for the Principal name for a SAML token.

Another sample mapping file that maps a SAML 1.x token to a local identity is sp_saml_1x_ext.xsl. This file performs the mapping as described but adds a section that verifies if the identity provider has used an acceptable level of authentication. In this sample file, an exception is thrown if the identity provider has used password authentication.

```
<xsl:param name="message">Detected an unacceptable authentication method.
  A higher level of authentication is required.</xsl:param>
 <xsl:template match="//stsuuser:AttributeList">
 <xsl:variable name="result" select="//stsuuser:AttributeList/
  stsuuser:Attribute[@name='AuthenticationMethod']/stsuuser:Value"/>
 <xsl:if test="(contains($result,'password')) = 'true'">
  <xsl:value-of select="mapping-ext:throwSTSException($message)" />
 </xsl:if>
 </xsl:template>
```

Figure 13. XSL code sample showing verification of a value for the AuthenticationMethod

Mapping a local identity to a SAML 2.0 token using an alias

This scenario occurs when messages are exchanged between partners in a SAML 2.0 single sign-on federation. When a user request is received (for example, for access to a remote resource) Tivoli Federated Identity Manager contacts the point of contact server (for example, WebSphere Application Server) and obtains a local user identity. The scenario described here uses the ip_saml_20.xsl sample mapping file in which an alias is used for the identity.

The Tivoli Federated Identity Manager server places the local user identity information into an XML document that conforms to the security token service universal user (STSUUSER) schema. The server then consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

The identity mapping module then modifies the XML document to contain the information required to build a SAML 2.0 token.

Table 21. STSUUSER entries used to generate a SAML token (using an alias)

STSUUSER element	SAML Token Information	Required?
Attribute: AuthContextClassRef	The authentication context class reference. Note that this element is always set to "password" (username/password) regardless of the authentication method that is set in the credential.	Required
Attribute List	Additional custom attributes	Optional

The mapping module is responsible for the following tasks:

1. Mapping Principal Attr Name to a Principal Name entry.

   Note that when the token module generates the token, this Principal name is not directly used. Instead, the value in the Name field is sent as input to the Tivoli Federated Identity Manager alias service. The alias service obtains the alias name (name identifier) for the principal, and places the returned alias in the generated token module.

   The type must be valid for SAML. For example:

   `urn:oasis:names:tc:SAML:2.0:assertion`
2. Setting the authentication method to the `password` mechanism. This action is required by the SAML standard.

   The following code sample shows part of the default mapping rule file, ip_saml_20.xsl.

```
<!--
Note: No Principal template is necessary for identity provider on SAML 2.0 since name identifiers
will be carried in the Subject element of the assertion.
-->

 <xsl:template match="//stsuuser:AttributeList">
 <stsuuser:AttributeList>
                       <!-- First the authentcation context class ref. attribute -->
  <stsuuser:Attribute name="AuthnContextClassRef" type="urn:oasis:names:tc:SAML:2.0:assertion">
   <stsuuser:Value>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:AttributeList>
 </xsl:template>
```

Figure 14. XSL code sample showing mapping of a local user identity into a SAML token (using an alias)

3. Populating the attribute statement of the assertion with the attributes in the AttributeList in the In-STSUU. This information becomes custom information in the token.

   There can be custom attributes that are required by applications that will make use of information that is to be transmitted between federation partners.

Mapping a SAML 2.0 token to a local identity

The service provider receives a SAML 2.0. Tivoli Federated Identity Manager converts the token contents into an STSUU document that conforms to the security token service universal user schema.

Table 22. SAML token information that is converted into a STS universal user document

SAML Token Information	STSUUSER element
AuthenticationStatement/Subject/NameIdentifier	Principal Attr: Name
Additional custom attributes	AttributeList (Optional)

The token module reads the token and obtains the NameIdentifier. The token module passes the NameIdentifier (an alias) to the alias service. The alias service converts the received alias to the local identity. The token module puts the local identity into the Principal element in the STSUU document.

- The NameIdentifier alias that is returned is used to populate the `name` attribute of the Principal This is the local user ID.

The following code example shows the assignment of a set value for the Principal name. This code sample is from the default mapping file sp_saml_20.xsl

```
<!--
  This will replace the principal name with the user's local name.
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
   <stsuuser:Value>
    <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value"/>
   </stsuuser:Value>
  </stsuuser:Attribute>
 </xsl:template>
```

Figure 15. XSL code sample showing assignment of a value for the Principal name for a SAML 2.0 token.

- Other information from the token is used to populate Attributes in the AttributeList.

  The following code example shows the optional assignment of additional values to attributes. This code sample is from the default mapping file sp_saml_20.xsl.

```
<xsl:variable name="department">
<xsl:value-of select="//stsuuser:AttributeList/stsuuser:Attribute[@name='Department']/stsuuser:Value"/>
 </xsl:variable>

<xsl:template match="//stsuuser:AttributeList">
 <stsuuser:AttributeList>
  <stsuuser:Attribute type="urn:ibm:names:ITFIM:5.1:accessmanager">
   <xsl:attribute name="Department">
    <stsuuser:Value>
  <xsl:value-of select="//stsuuser:AttributeList/stsuuser:Attribute[@name='Department']
  /stsuuser:Value"/>
    </stsuuser:Value>
   </stsuuser:Attribute>
 </stsuuser:AttributeList>
</xsl:template>
```

Figure 16. XSL code sample showing AttributeList for a SAML 2.0 token.

Chapter 16. SAML 2.0 Attribute query

The SAML 2.0 attribute query feature extends the capability of the SAML 2.0 protocol. Traditional SAML 2.0 function requires that the identity provider sends the federation partner *all* required user attributes. The attributes are included as part of the assertion generated during the single sign-on flow.

The SAML 2.0 attribute query feature eliminates this limitation. Administrators for identity providers can include in the single sign-on flow only the attributes that are used by most targeted applications. Applications can use a SAML 2.0 attribute query flow to obtain any attribute requirements or specialized values.

Support for attribute query provides a set of core attributes when the initial authentication context is established. You can query user information as needed during the application runtime operation. Different applications require different user information. For example, applications that require fine grained authorization require specific user entitlements to make the authorization decisions.

Attribute query supports the following modes:

Direct mode
The requesting application issues a direct call to the identity provider to obtain any required attributes.

On-behalf mode
The requesting application contacts the service provider, which proxies the attribute request to the identity provider.

Direct mode

In direct mode, the requesting application sends an AttributeQuery request to the SAML 2.0 federation SOAP endpoint on the identity provider. The SOAP delegate protocol finishes the necessary protocol actions and issues a SAML assertion. The SAML attribute query function uses the attribute query secure token service (STS) module to issue the assertion.

The direct mode requires the application (attribute requester) to be known to the identity provider. To make an application known at the identity provider, you must import the requester metadata using the command-line interface command `manageItfimPartner`.

The single sign-on flow for direct mode is:

1. User requires access to a resource or application and initiates a federated single sign-on flow.
2. The identity provider authenticates the user and issues a SAML assertion with a subset of attributes that most applications or resources require.
3. The application or resource determines if any additional attributes are required. If so, the application issues an `AttributeQuery` to the identity provider obtain them.
4. The identity provider returns a SAML assertion with the requested attributes.
5. The application or resource obtains the attributes returned by the identity provider in the attribute query SAML response message.

On-behalf mode

On-behalf mode requires that applications send query requests to the service provider, which then proxies them to the identity provider. The identity provider supplies the requested attributes. On-behalf mode supports two different types of requests:

- SAML 2.0 <*AttributeQuery*> requests

  The application must send AttributeQuery messages to the service provider SOAP endpoint. If an AttributeQuery request message is used, the service provider returns a SAML Response message with the corresponding assertion.
- WS-Trust Request Security Token messages.

  For this protocol, the application must send WS-Trust messages to the trust service endpoint. If the requesting application sends a WS-Trust message, the response message is a Universal User Token

  Note: If your application is a WS-Trust client, you can use this option instead of using the SAML protocol.

The on-behalf mode limits the amount of configuration required at the identity provider for many service provider applications to query user attributes. In this mode, the service provider is the only known entity at the identity provider.

The single sign-on flow for on-behalf mode is:

1. User requires access to a resource or application on the service provider and initiates a federated single sign-on flow.
2. The identity provider authenticates the user and issues a SAML assertion with a subset of attributes that most applications or resources require.
3. The service provider selects which attributes to make available to the resource or application. The service provider then creates the authenticated session for the user.
4. The application or resource determines if any additional attributes are required. If so, the application issues an `AttributeQuery` or a WS-Trust `RequestSecurityToken` to obtain them. The application sends the request to the service provider. The service provider proxies the request to the identity provider.
5. The Identity Provider returns a SAML assertion with the requested attributes.
6. The application or resource obtains the attributes returned by the Identity Provider in the attribute query SAML response message. If a WS-Trust request is used, the attributes are returned to the client application using a Universal User Token. If the request is a SAML AttributeQuery request, the attributes are returned in a SAMLResponse generated by the Service Provider.

Attribute query request partner

The Attribute query feature defines a new type of role. Application partners to a SAML 2.0 federation can now act in an *attribute query requester* role. This role is different from the role of service provider partner or identity provider partner.

An attribute query requester is an entity that makes SOAP based `<AttributeQuery>` request calls to obtain user attributes.

If you plan to configure an *attribute query requester partner*, you must generate a metadata file as specified by the SAML 2.0 specification. Tivoli Federated Identity

Manager uses this metadata file to create the attribute request partner. You must use the manageItfimPartner command to create the partner. This command uses a response file, which contains a parameter that specifies the location of the metadata file.

Developing an attribute query STS module

The attribute query function uses an STS token module called the *attribute query module*. You must configure the module for the STS trust chain for the SAML 2.0 federation.

Before you configure attribute query, you must:

1. Determine the attributes that your resource or application wants to request from the identity provider.
2. Develop a script or module that requests the attributes. This request can be made by an XSLT or JavaScript file, a Tivoli Directory Integrator assembly line, or a custom secure token service (STS) mapping module.

Limitation with migrating from previous release of Tivoli Federated Identity Manager

Tivoli Federated Identity Manager supports migration of SAML 2.0 federations from the previous release to the current release. The attribute query feature was not available in previous releases. This means that when you migrate SAML 2.0 federations from the previous release, attribute query is not automatically enabled in the new release.

To enable attribute query for the federation, take the following steps after you have migrated the federation:

- Select the check box on the federation properties page to enable attribute query.
- Use the graphical user interface on the federation properties page to configure an attribute query module.
- Use the Add Partner wizard to add all partners that previously existed for the federation.

Configuring attribute query

You can configure SAML 2.0 federations and partners to support the attribute query feature.

The steps for attribute query configuration vary depending on the deployment scenario. Deployment includes the creation of a federation and the addition of a partner to the federation.

When you configure the federations, the identity provider partners, and the service provider partners, you can use a graphical user interface that prompts for attribute query settings. The section provides detailed descriptions of these settings.

Some settings for attribute query use existing values for SAML 2.0 federations. For these settings, you are not prompted for additional configuration for attribute query.

For example, providers sign or validate assertions based on the configuration settings established for the SAML 2.0 federation or partner. The federation or partner signs or validates the attribute query assertions as required by the federation partner. You are not required to specify additional settings to enforce signing or validation.

If you install SAML 2.0 with the Typical or All profiles, signing and validation are activated automatically. If you select manual installation of profiles, the wizard prompts you to specify whether to sign and validate messages. The wizard requires these settings whether the attribute query feature is configured or not configured.

To configure your federation and partner in direct mode, complete the following tasks:

- "Creating a federation as an attribute authority"
- "Creating an attribute query request partner" on page 166

To configure your federation and partner in on-behalf mode, complete the following tasks:

- "Creating a federation as an attribute authority"
- "Creating an identity provider partner or service provider partner for an attribute authority federation" on page 164
- "Creating an attribute query request partner" on page 166

Creating a federation as an attribute authority

You can use either the administration console or the command-line interface to create a SAML 2.0 federation as an attribute authority.

Choose one of the following methods:

- "Using the administration console to create a federation as an attribute authority"
- "Using the command line interface to create a federation as an attribute authority" on page 163

Using the administration console to create a federation as an attribute authority

You can use the administration console to create a SAML 2.0 federation as an attribute authority.

About this task

The configuration for attribute query uses the same wizard as is used for all SAML federations. When you use the wizard, you activate attribute query, and are prompted to provide configuration settings.

Parameters for attribute query are described in the worksheets for federation configuration. See the topic for your partner type:

- "SAML 2.0 identity provider worksheet" on page 178.
- "SAML 2.0 service provider worksheet" on page 174.

Note: Combine the information in this procedure below with the step by step configuration instructions for SAML 2.0 federations in Chapter 17, "Establishing a SAML federation," on page 169.

Procedure

1. On the Profiles panel in the wizard, select **All** or **Manual**.
2. On the Profile Details panel, go to Attribute Query and select **Enabled**. Selection of this check box causes additional panels to be shown.
3. On the SAML Assertions panel, specify the amount of time before the issue date that an assertion is considered valid. Specify also the amount of time that the assertion is valid after being issued.

   Note: When you are using a *service provider* federation for attribute query, the federation must issue assertions. This requirement means that when you have activated attribute query for a service provider federation, the SAML assertions panel displays and you must specify values. When you configure a service provider federation without attribute query, you are not required to set values for SAML assertions.

   The SAML Assertions panel is shown for *identity provider* federation creation regardless of whether attribute query is selected. In this type of federation, SAML assertions are issued for multiple purposes.
4. On the Attribute Module Selection panel, select one of the following choices:
   - XSLT or JavaScript transformation
   - Tivoli Directory Integrator module
   - Custom mapping module.

   Base your selection on the method you identified for your deployment when you planned the configuration.

Using the command line interface to create a federation as an attribute authority

You can use the command-line interface to create a SAML 2.0 federation as an attribute authority.

About this task

When using the command-line interface to create a SAML 2 federation, you must first create and populate a SAML 2 federation response file. To establish the SAML 2 federation as an attribute authority, you must set values in the response file for the following parameters:

- `AttributeQueryMappingRule`
- `AttributeQueryMappingRuleFileName`
- `AttributeAuthorityEnabled`
- `SignAttributeQueryRequest`
- `SignAttributeQueryResponse`

For descriptions of the parameters needed, see "SAML 2.0 attribute query federation response file parameters" on page 167

For more information about using the command-line interface to create a SAML 2 federation and a response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a SAML 2 response file.

   For example, to create a SAML 2 response file based on an existing federation:

   ```
   $AdminTask manageItfimFederation {-operation createResponseFile
   -fimDomainName domain1 -federationName idpsaml2
   -fileId c:\temp\saml2idp.rsp }
   ```
2. Edit the SAML 2 response file to set the attribute query parameters.

   In the example, the response file is `c:\temp\saml2idp.rsp`
3. Create the SAML 2 federation as an attribute authority.

   To create an identity provider or service provider federation that is activated for attribute query, use the standard syntax. There are no additional options to specify.

   For example, if the response file is `c:\temp\saml2idp.rsp`:

   ```
   $AdminTask manageItfimFederation { -operation create -fimDomainName domain1
   -fileId c:\temp\saml2idp.rsp }
   ```

Creating an identity provider partner or service provider partner for an attribute authority federation

You can create an identity provider partner or service provider partner for a SAML 2.0 federation that has been configured as an attribute authority.

When a federation has been configured to as an attribute authority, you can add partners of the following types:

- Service provider partner

  Add a traditional service provider partner to an identity provider federation. You can configure this partner to exchange attribute query request-responses with the federation provider.
- Identity provider partner

  Add a traditional identity provider partner to a service provider federation. You can configure this partner to exchange attribute query request-responses with the federation provider.
- Attribute query request partner

  This type of partner is a special case for use when the requesting application or resource does not have Tivoli Federated Identity Manager installed.

  Note: The instructions in this topic do not apply to attribute query request partners. See "Creating an attribute query request partner" on page 166.

To add either an identity provider partner or a service provider partner, see:

- "Using the administration console to create a service provider or identity provider partner"
- "Using a command-line interface to create a service provider or identity provider partner" on page 166

Using the administration console to create a service provider or identity provider partner

You can use the administration console to create a service provider or identity provider partner.

About this task

You can use the Add Partner wizard to add a service provider partner or identity provider partner to a federation. This wizard is also used for adding SAML 2.0 partners without attribute query.

When you use the wizard to add a partner to a federation, the configuration program determines if the federation is configured as an attribute query authority. If the federation is an attribute query authority, additional panels prompt you to enter more information.

The configuration panels differ slightly for identity provider partners or service provider partners. See the following table.

Configuration panel	Partner type	Description
SAML Assertions	Identity provider partner for a service provider federation *only*	The SAML Assertions settings panel permits you to specify which attributes to include in the assertion. The default value is to include all attributes. You can use this setting to specify a base set of attributes. The SAML Assertions panel also permits you to specify which attributes get encrypted, and which encryptions algorithms to use.
Attribute Module Selection	Both identity provider partner for a service provider federation and service provider partner for an identity partner federation	On the Attribute Module Selection panel, you must select one of: • XSLT or JavaScript transformation • Tivoli Directory Integrator module • Custom mapping module. Base your selection on the method you identified for your deployment when you planned the configuration.

The parameters for partner configuration for attribute query are described in the worksheets for SAML 2.0 partner configuration. See the topic for your partner type:

- "SAML 2.0 identity provider partner worksheet" on page 203
- "SAML 2.0 service provider partner worksheet" on page 198

The graphical user interface wizard for adding SAML 2.0 partners includes the panels for attribute query configuration. To configure the identity provider or service provider partner, see the SAML 2.0 instructions: "Adding your partner" on page 209

Using a command-line interface to create a service provider or identity provider partner

You can create an identity provider partner or service provider partner for a SAML 2.0 federation that has been configured as an attribute authority.

About this task

When using the command-line interface to create a partner, you must first create and populate a SAML 2 partner response file. To configure the partners to use the attribute query capability, you must set values for the following parameters in the response file:

- `AttributeQueryMappingRule`
- `AttributeQueryMappingRuleFileName`
- `ValidateAttributeQueryRequest`
- `ValidateAttributeQueryResponse`

For information about using the command-line interface to create a SAML 2 partner and partner response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a SAML 2 partner response file.

   For example, to create a SAML 2 partner response file based on an existing partner:

   ```
   $AdminTask manageItfimPartner {-operation createResponseFile
   -fimDomainName domain1  -federationName fed1
   -partnerName idppartner -fileId c:\temp\saml2idp.rsp }
   ```

2. Edit the SAML 2 partner response file to set the attribute query parameters.

   In the example, the response file is `c:\temp\saml2idp.rsp`

   For descriptions of the attribute query response file parameters, see "SAML 2.0 attribute query partner response file parameters" on page 168

3. To create an identity provider partner that is configured for attribute query, use the standard syntax.

   You can optionally specify the partner role in the command line. You are not required to specify the partner role. When the role is not specified the program automatically sets the partner role based on the federation role.

   For example, if the response file is `c:\temp\saml2idp.rsp`:

   ```
   $AdminTask manageItfimPartner { -operation create -fimDomainName domain1
   -federationName idpsaml2 -partnerName idpartner
   -fileId c:\temp\saml2idp.rsp
   -signingKeystorePwd testonly -encryptionKeystorePwd testonly }
   ```

   If you want to specify the partner role in the command line, add the `-partnerRole` option, and specify either `sp` or `idp`. For example, to specify a service provider partner:

   ```
   $AdminTask manageItfimPartner { -operation create -fimDomainName domain1
   -federationName idpsaml2 -partnerName idpartner
   -partnerRole sp
   -fileId c:\temp\saml2sp.rsp
   -signingKeystorePwd testonly -encryptionKeystorePwd testonly }
   ```

Creating an attribute query request partner

Use the command-line interface to create an attribute query request partner.

About this task

You must use the command-line interface to add an attribute query request partner to a federation. The administration graphical user interface does not provide a wizard for this task.

Use the manageItfimPartner command to create the partner. This command supports a partner role parameter qr that indicates that a query requester partner is to be created.

Procedure

1. Create a SAML 2 partner response file.

   For example, to create a SAML 2 attribute query request partner response file based on an existing partner:

   ```
   $AdminTask manageItfimPartner { -operation createResponseFile
   -fimDomainName fimipdomain  -federationName saml20ip
   -partnerRole qr -fileId /downloads/qr.out }
   ```
2. Edit the response file to show the location of the metadata file from the attribute query request partner. This file name is a parameter in the response file. You also must add information specific to the partner.

   For information about using the command-line interface to create a SAML 2 partner and partner response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.
3. Create an attribute requester partner:

   ```
   $AdminTask manageItfimPartner { -operation create
   -fimDomainName fimipdomain
   -federationName saml20ip -partnerName samlqr
   -partnerRole qr -fileId /downloads/qr.out
   -signingKeystorePwd testonly
   -encryptionKeystorePwd testonly}
   ```

SAML 2.0 attribute query federation response file parameters

The SAML 2.0 federation response file contains parameters that are used by attribute query.

Table 23. Attribute query parameters for federation response file

Parameter	Value	Description
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that are used to format the rule, so that it can be contained in the XML response file. Use this property to specify a mapping rule without using a file on the file system. Use this property also if you are modifying a federation. If you want to edit the XSLT rule as a regular file, supply it to the response file using the **AttributeQueryMappingRuleFileName** property. This rule is used for attribute query operations.
AttributeQueryMappingRuleFileName	*path and file name*	Specifies path name to an XSLT file that is used as a mapping rule. When defined, it takes precedence over the **AttributeQueryMappingRule** property. This rule is used for attribute query operations.

Table 23. Attribute query parameters for federation response file (continued)

Parameter	Value	Description
AttributeAuthorityEnabled	*true or false*	Specifies whether the attribute query feature is configured in the federation. The value `true` activates attribute query. The value `false` disables attribute query. Default: false
SignAttributeQueryResponse	*true or false*	Specifies whether attribute query responses are signed.
SignAttributeQueryRequest	*true or false*	Specifies whether attribute query requests are signed.

SAML 2.0 attribute query partner response file parameters

The SAML 2.0 partner response file contains parameters that are used by attribute query.

Table 24. Attribute query parameters for partner response file

Parameter	Value	Description
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that are used to format the rule, so that it can be contained in the XML response file. Use this property if you want to specify a mapping rule without using a file on the file system. Use this property also if you are modifying a federation. If you want to edit the XSLT rule as a regular file, supply it to the response file using the **AttributeQueryMappingRuleFileName** property. This rule is used for attribute query operations.
AttributeQueryMappingRuleFileName	*path and file name*	Specifies path name to an XSLT file that is used as a mapping rule. When defined, it takes precedence over the **AttributeQueryMappingRule** property. This rule is used for attribute query operations.
ValidateAttributeQueryResponse	*true or false*	Specifies that validation of the partner signatures takes places on attribute query responses that are received. An error is shown if the message is not signed.
ValidateAttributeQueryRequest	*true or false*	Specifies that an attribute query request that was received from the partner signature is validated. An error is shown if the message is not signed.

Chapter 17. Establishing a SAML federation

Complete the following tasks to configure your federation:

1. "Gathering your federation configuration information."
2. "Creating your role in the federation" on page 184.
3. "Providing guidance to your partner" on page 186.
4. "Obtaining federation configuration data from your partner" on page 188.
5. "Adding your partner" on page 209.
6. "Providing federation properties to your partner" on page 210.

Gathering your federation configuration information

The Federation wizard prompts you for information that is used in your federation. Before starting the wizard, prepare for the configuration process by gathering your configuration information using the appropriate worksheet.

About this task

Choose a worksheet based on the SAML standard that you want to use in the federation and your role in the federation.

- "SAML 1.x service provider worksheet"
- "SAML 1.x identity provider worksheet" on page 171
- "SAML 2.0 service provider worksheet" on page 174
- "SAML 2.0 identity provider worksheet" on page 178

SAML 1.x service provider worksheet

If you will be the service provider in the federation and will use SAML 1.0 or SAML 1.1, record your configuration information in the following tables.

Table 25. General information for service provider in SAML 1.x federation

General Information	Description	Your value
Federation name	The unique name you will give to the federation.	
Role	The role you will provide in the federation. (In these instructions, you are the service provider.)	Service provider

Table 26. Contact information for service provider in SAML 1.x federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and optionally other information about the contact associated with your role in the federation.	

Table 27. Federation protocol for service provider in SAML 1.x federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner will use in the federation.	One of the following: • SAML 1.0 • SAML 1.1

Table 28. Point of contact server information for service provider in SAML 1.x federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 29. Signature information for service provider in SAML 1.x federation

Signatures	Description	Your value
Sign Artifact Resolution Requests	A check box that indicates that you will sign request messages. Default value: No signing. The check box is not selected.	One of the following: • Sign request messages. (Select check box.) • Do not sign request messages. (Clear check box.)
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign request messages	If you select the check box, you must supply the signing key that you will use to sign the requests. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, “Setting up message security,” on page 29.	Keystore name: Keystore password: Key alias name:

Table 30. Identity mapping information for service provider in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation (XSLT) file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation. .	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 184.

SAML 1.x identity provider worksheet

If you will be the identity provider in the federation and will use SAML 1.0 or SAML 1.1, record your configuration information in the following tables.

Table 31. General information for identity provider in SAML 1.x federation

General Information	Description	Your value
Federation name	The unique name you will give to the federation.	
Role	The role you will provide in the federation. (In these instructions, you are the identity provider.)	Identity provider

Table 32. Contact information for identity provider in SAML 1.x federation

Contact Information	Description	Your values
Company name, Company URL, and contact name and information.	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 33. Federation protocol information for identity provider in SAML 1.x federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner will use in the federation.	One of the following: • SAML 1.0 • SAML 1.1

Table 34. Point of contact server for identity provider in SAML 1.x federation

Point of Contact Server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 35. Signing information for identity provider in SAML 1.x federation

Signatures	Description	Your value
Signature options: • **SAML messages for Browser POST profile are signed** (required) • **Sign SAML messages for artifact profile** (optional)	• When browser POST is used as the profile, SAML messages must be signed. Therefore, it is pre-selected and cannot be deselected. • You have the option of also signing the SAML messages when browser artifact is used.	One of the following: • Sign browser artifact messages. (Select check box.) • Do not sign browser artifact messages. (Clear check box.)
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use for signing	Because Browser POST messages must be signed, you are required to supply a signing key. If you select to also sign messages when browser artifact is used, the same signing key will be used to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, “Setting up message security,” on page 29.	Keystore name: Keystore password: Key alias name:

Table 36. SAML Message Settings information for identity provider in SAML 1.x federation

SAML Message Settings	Description	Your value
Artifact Resolution Service URL	The URL for your artifact resolution endpoint. (**Note:** The value for this field is filled in automatically using the point of contact server URL you specified earlier.)	
Artifact Cache Lifetime (seconds)	The artifact cache lifetime in seconds. Default value: 30 seconds.	

Table 36. SAML Message Settings information for identity provider in SAML 1.x federation (continued)

SAML Message Settings	Description	Your value
Allow IBM Protocol Extension	You must specify whether you will allow the use of the IBM PROTOCOL extension. The extension allows a query-string parameter that specifies whether browser artifact or browser POST will be used. For more information, see "SAML 1.x" on page 137.	One of the following: • Allow IBM Protocol Extension. (Select the check box.) • Do not allow Protocol Extension. (Clear the check box.)

Table 37. Token Settings information for identity provider in SAML 1.x federation

Configure Token Settings	Description	Your value
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion will be considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion will be considered valid after its issue date. Default value: 60	

Table 38. Identity mapping information for identity provider in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation (XSLT) file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 184.

SAML 2.0 service provider worksheet

If you will be the service provider in the federation and will use SAML 2.0, record your configuration information in the following tables.

Table 39. General information for service provider in SAML 2.0 federation

General Information	Description	Your value
Federation name	The unique name you will give to the federation.	
Role	The role you will provide in the federation. (In these instructions, you are the service provider.)	Service provider

Table 40. Contact information for service provider in SAML 2.0 federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and optionally other information about the contact associated with your role in the federation.	

Table 41. Federation protocol for service provider in SAML 2.0 federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner will use in the federation.	SAML 2.0

Table 42. Point of contact server information for service provider in SAML 2.0 federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 43. Profile selection and configuration information for service provider in SAML 2.0 federation

Profile selection	Description	Your value
SAML 2.0 profile options: Choose one of the following profile options:	The profile for your federation. For more information about profiles, see “SAML 2.0” on page 139.	One of the following: • Basic • Typical • All • Manual
Basic: Web Browser SSO, Single Logout	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout	(No additional values required.)

Table 43. Profile selection and configuration information for service provider in SAML 2.0 federation (continued)

Profile selection	Description	Your value
Typical: Web Browser SSO, Single Logout and Name Identifier Management	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout • Enhanced client or proxy • Name Identifier Management	(No additional values required.)
Enable all profiles and bindings	If you choose **Enable all profiles and bindings**, you must be ready to provide the following information on subsequent panels: **Identity Provider Discovery Settings panel:** • Common domain name • Common domain cookie service URL **Enhanced Client Proxy panel:** • HTTP headers	**Identity Provider Discovery Settings** • Common domain name: • Common domain cookie service URL: **Enhanced Client Proxy** HTTP headers:
Manual: Choose individual profiles and bindings	If you choose **Manual**, you must be ready to select individual profiles and supported bindings.	Profiles and bindings:

Table 44. Signature information for service provider in SAML 2.0 federation

Signatures	Description	Your value
Require signature on incoming messages and assertions	A check box that specifies that your partner will use its private key to sign the message and assertion. Default value: The check box is checked.	One of the following: • Partner will sign. (Check box is selected.) • Partner will not sign. (Check box is not selected.)
Select which outgoing messages and assertions you will sign	Buttons that indicate which outgoing messages you will sign. The default setting is for the typical set of outgoing SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of outgoing SAML messages are signed. • All outgoing SAML messages and assertions are signed. • No outgoing SAML messages and assertions are signed.

Table 44. Signature information for service provider in SAML 2.0 federation (continued)

Signatures	Description	Your value
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign messages	If you will sign messages and assertions, you must supply the signing key that you will use to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, "Setting up message security," on page 29.	Keystore name: Keystore password: Key alias name:

Table 45. Encryption information for service provider in SAML 2.0 federation

Encryption	Description	Your value
Encryption Key: • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Public/private key pair that will be used for data you receive from your partner.	A public/private key pair used in encryption. Your partner will use the public key to encrypt data to you. You will use the private key to decrypt data that your partner sends to you. You must specify the key pair that you will use. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. .	Keystore name: Keystore password: Key alias name:

Table 46. SAML message settings for service provider in SAML 2.0 federation

Message settings	Description	Your value
Message Options: • Message Lifetime in seconds • Artifact Lifetime in seconds • Session Timeout	Amount of time in seconds that messages, artifacts, and sessions are valid. The default values are: • Message lifetime: 300 • Artifact lifetime: 120 • Session timeout: 7200	Message Lifetime in seconds: Artifact Lifetime in seconds: Session Timeout:
Single sign-On Options • Identity Provider is allowed to interact with user • Single sign-on is passive • Force Identity Provider to authenticate user	Specifies how the identity provider is to interact with the users.	One of the following: • Identity Provider is allowed to interact with user • Single sign-on is passive • Force Identity Provider to authenticate user

Table 46. SAML message settings for service provider in SAML 2.0 federation (continued)

Message settings	Description	Your value
SOAP Endpoint	The URL of the SOAP endpoint. Default value: The value in this field is based on the point of contact server URL that you supplied earlier. **Note:** If the SOAP binding is not used in the profile you selected, this field is not displayed.	

Table 47. Attribute query information for service provider

Attribute query	Description	Your value
Enabled	Indicates if the provider is permitted to act as the attribute authority. If the check box is selected, the attribute query profile is activated.	
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion is considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion is considered valid after its issue date. Default value: 60	

Table 48. Attribute query mapping information for service provider in SAML 2.0 federation

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you must have the file created before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 49. Identity mapping information for service provider in SAML 2.0 federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in “Creating your role in the federation” on page 184.

SAML 2.0 identity provider worksheet

If you will be the identity provider in the federation and will use SAML 2.0, record your configuration information in the following tables.

Table 50. General information for identity provider in SAML 2.0 federation

General Information	Description	Your value
Federation name	The unique name you will give to the federation.	
Role	The role you will provide in the federation. (In these instructions, you are the identity provider.)	Identity provider

Table 51. Contact information for identity provider in SAML 2.0 federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and optionally other information about the contact associated with your role in the federation.	

Table 52. Federation protocol for identity provider in SAML 2.0 federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner will use in the federation.	SAML 2.0

Table 53. Point of contact server information for identity provider in SAML 2.0 federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 54. Profile selection and configuration information for identity provider in SAML 2.0 federation

Profile selection	Description	Your value
SAML 2.0 profile options: Choose one of the following profile options:	The profile for your federation. For more information about profiles, see "SAML 2.0" on page 139.	One of the following: • Basic • Typical • All • Manual
Basic: Web Browser SSO, Single Logout	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout	(No additional values required.)

Table 54. Profile selection and configuration information for identity provider in SAML 2.0 federation (continued)

Profile selection	Description	Your value
Typical: Web Browser SSO, Single Logout and Name Identifier Management	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout • Enhanced client or proxy • Name Identifier Management	(No additional values required.)
Enable all profiles and bindings	If you choose **Enable all profiles and bindings**, you must be ready to provide the following information on subsequent panels: **Identity Provider Discovery Settings panel** • Common domain name • Common domain cookie service URL • Common domain cookie lifetime in seconds. Default value: 1 **Enhanced Client Proxy panel** • HTTP headers	**Identity Provider Discovery Settings panel** • Common domain name • Common domain cookie service URL • Common domain cookie lifetime in seconds. Default value: 1 **Enhanced Client Proxy panel** • HTTP headers
Manual: Choose individual profiles and bindings	If you choose **Manual**, you must be ready to select individual profiles and supported bindings.	Profiles and bindings:

Table 55. Signature information for identity provider in SAML 2.0 federation

Signatures	Description	Your value
Require signature on incoming messages and assertions	A check box that specifies that your partner will use its private key to sign the message and assertion. Default value: The check box is checked.	One of the following: • Partner will sign. (Check box is selected.) • Partner will not sign. (Check box is not selected.)
Select which outgoing messages and assertions you will sign	Buttons that indicate which outgoing messages you will sign. The default setting is for the typical set of outgoing SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of outgoing SAML messages are signed. • All outgoing SAML messages and assertions are signed. • No outgoing SAML messages and assertions are signed.

Table 55. Signature information for identity provider in SAML 2.0 federation (continued)

Signatures	Description	Your value
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign messages	If you will sign messages and assertions, you must supply the signing key that you will use to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task.	Keystore name: Keystore password: Key alias name:

Table 56. Encryption information for identity provider in SAML 2.0 federation

Encryption	Description	Your value
Encryption Key: • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Public/private key pair that will be used for data you receive from your partner.	A public/private key pair used in encryption. Your partner will use the public key to encrypt data to you. You will use the private key to decrypt data that your partner sends to you. You must specify the key pair that you will use. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, “Setting up message security,” on page 29.	Keystore name: Keystore password: Key alias name:

Table 57. SAML message settings for identity provider in SAML 2.0 federation

Message settings	Description	Your value
Message Options: • Message Lifetime in seconds • Artifact Lifetime in seconds • Session Timeout	Amount of time in seconds that messages, artifacts, and sessions are valid. The default values are: • Message lifetime: 300 • Artifact lifetime: 120 • Session timeout: 7200	Message Lifetime in seconds: Artifact Lifetime in seconds: Session Timeout:
Require Consent to Federate	If you select this check box, you are required to present a page to the user to verify that the user has made a federation request. Default value: Requires consent to federate.	One of the following: • Require Consent to Federate (Check box is selected.) • Do not require consent to federate. (Check box is not selected.)

Table 57. SAML message settings for identity provider in SAML 2.0 federation (continued)

Message settings	Description	Your value
SOAP Endpoint	The URL of the SOAP endpoint. Default value: The value in this field is based on the point of contact server URL that you supplied earlier. **Note:** If the SOAP binding is not used in the profile you selected, this field is not displayed.	

Table 58. Token Settings information for identity provider in SAML 2.0 federation

Configure Token Settings	Description	Your value
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion will be considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion will be considered valid after its issue date. Default value: 60	

Table 59. Attribute query information for identity provider

Attribute query	Description	Your value
Enabled	Indicates if the provider is permitted to act as the attribute authority. If selected, the attribute query profile is activated.	

Table 60. Attribute query mapping information for identity provider

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you must have the file created before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 61. Identity mapping information for identity provider in SAML 2.0 federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 184.

Creating your role in the federation

Use the console to create a federation. To begin, the Federation Wizard prompts you to supply the necessary information about your role in the federation. For descriptions of the fields that you are prompted for by the wizard, refer to the online help.

Before you begin

Before beginning this procedure, complete the worksheet that is appropriate for the SAML standard you will use and for your role in the federation:

- "SAML 1.x service provider worksheet" on page 169
- "SAML 1.x identity provider worksheet" on page 171
- "SAML 2.0 service provider worksheet" on page 174
- "SAML 2.0 identity provider worksheet" on page 178

About this task

Note: During the configuration, you might be prompted to restart WebSphere Application Server. Make sure the server has restarted completely before continuing with the task.

To create a federation:

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The General Information panel is displayed.
3. Use your worksheet to complete the panels that are displayed by the Federation wizard. Use your completed worksheet as a guide for completing the fields that are displayed. If you need to go back to a previous panel, click **Back**. If you want to end the configuration, click **Cancel**. Otherwise, click **Next** after you complete each panel.
4. When you have completed all configuration panels, the Summary panel is displayed. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.
5. You can add your partner now or later. Choose one:
   - Click **Add partner** to start the Partner Wizard and add your partner's configuration using the steps described in:
     a. "Obtaining federation configuration data from your partner" on page 188, including complete the appropriate worksheet for your partner's role in the federation.
     b. "Adding your partner" on page 209.
   - To add your partner at a later time, click **Done**. You will return to the Federations panel.

Configuring a WebSEAL point of contact server for the SAML federation

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the SAML federation.

Before you begin

The federation wizard provides a button that you can use to obtain a configuration utility.

About this task

You must obtain the utility and run it. Complete the following steps:

Procedure

1. Click **Download Tivoli Access Manager Configuration Tool**
2. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
3. Return to the management console, and Click **Done** to return to the Federations panel.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation we will complete other tasks first.
4. Run the configuration tool from a command line. The syntax is:
   ```
   java -jar /download_dir/tfimcfg.jar -cfgfile webseald-instance_name.conf
    -action tamconfig
   ```
   You will need to know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -cfgfile webseald-default -action tamconfig
```

For more information, see:

- Appendix A, “tfimcfg reference,” on page 491

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log in to the administration console.
2. Click Tivoli Federated Identity Manager > Manage Configuration > Point of Contact
3. Select **WebSphere**

4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Providing guidance to your partner

When you are working with partners to establish a federation, you will need to provide information to them and collect information from them.

Depending on the role you will perform in your federations, you might find that you will need to give guidance or assistance to your partner, in addition to providing configuration information. The experience of your partner can help you decide the best way to provide guidance. Partners who have experience with single sign-on might need limited guidance, such as through phone calls or e-mail. However, partners who are new to single sign-on might need an orientation, such as through a tutorial or written description.

The time at which you provide guidance is up to you. You might want to provide it at the same time you solicit information from your partner. Or, you might want to share introductory information in the early stages of your federated relationship, before any configuration has taken place.

Use the outline below to help you prepare a guidance document for your partner. The outline assumes that you are the partner who is providing guidance; however, if you are the partner who needs guidance, consider providing the outline to your partner or modifying the outline into a questionnaire that you can use to request information from your partner.

Integration Guide outline

I. Introduction

a. Explain what single sign-on is and consider explaining your use of Tivoli Federated Identity Manager.

b. Define terminology such as federation, identity provider, service provider, and possibly protocol, profile, and binding.

c. Identify which role you will play and which your partner will play in the federation.

d. Describe how the end users will interact with your site and with your partner's site. For example, identify the service that the end users are trying to access. Consider including a graphic that identifies the flow of activities among the participants such as the end user signing on, the identity provider authenticating the user, the service provider granting access, and the end user accessing the service.

II. Technical specifications

a. State requirements or options for protocol, binding, and profile. For example, perhaps you require your partner to use SAML 1.1 with Browser Artifact. Or perhaps you require your partner to use SAML 1.1 but the profile type is the partner's choice.

b. Explain assertion requirements or options. For example, you might require that the partner include specific fields in the assertion, such as a user group mapping key with an individual identifier. Or, you might need to explain that assertion options will need to be specified such as assertion lifetime, artifact lifetime (if using Browser Artifact), and signing information.

c. Present any limitations about the types of devices that can use the single sign-on function. For example, the federation might support only Web browser interaction from end users.

d. Describe auditing and logging requirements. For more information, refer to the *IBM Tivoli Federated Identity Manager Auditing Guide*.

e. Explain how end users will experience event messages when interacting with the federation. For example, if you are a service provider you might provide customization options to your partner for how end users will log out or receive system messages about timeouts or other events. If you are an identity provider, you might provide customization options to your partner for how its end users will log in.

f. Agree how you and your partner will synchronize your system clocks.

III. Security

a. State SSL requirements.

b. Request certificate information (such as the name of the certificate authority that issued the partner's certificate or a copy of the partner's certificate).

c. Explain signing requirements or options.

IV Data exchange

Establish how federation data, including keys will be exchanged. (In SAML 1.x federations, data can be exchanged through a metadata file or manually. In SAML 2.0, a metadata file must be used.) If a manual method is used, list the information that you will require from your partner. Use the worksheets in “Obtaining federation configuration data from your partner” on page 188 and “Providing federation properties to your partner” on page 210.

V Testing

Explain your capability for testing the federation and include any requirements that your partner must follow before using the federation in a production environment. Consider including the URLs that are needed by your partner for testing purposes. For example, if you are the service provider in a SAML 1.x federation, you might need to provide a target URL and assertion consumer URL to your partner.

VI. Production

Explain what conditions must be met before the federation is ready for production. You might provide production URLs or explain how you will provide those URLs at a later time.

VII. Support

Explain how end user or administrator support will be handled in the federation.

VIII. Partner worksheet

At various points throughout the preceding sections, you might have requested information from your partner or explained why you would be requesting that information. At the end of your document, consider adding a worksheet where your partner can record that requested information. The worksheet might contain fields such as:

- Endpoint URLs for testing
- Endpoint URLs for production
- Contact information
- SSL certificate information (name of the certificate authority and so on)
- Signing information (what will be signed, what must be validated, and so on)
- Data exchange method (manual or metadata). If a manual method is used, you might need to add other fields to the worksheet to request the needed information.

Obtaining federation configuration data from your partner

You must obtain configuration information from your partner before you can add that partner to a federation.

The partner can export the federation configuration to a metadata file or, if the partner is using SAML 1.x, the partner can manually communicate the federation configuration to you. (Configuring partners manually is not supported in SAML 2.0 federations.)

To help you gather the appropriate information from your partner, complete the appropriate worksheet for the SAML standard you will use in the federation and for the role that your partner will have in the federation:

- *If you are the identity provider*, you will add a service provider partner. Use the service provider partner worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x service provider partner worksheet"
  - "SAML 2.0 service provider partner worksheet" on page 198
- *If you are the service provider*, you will add an identity provider partner. Use the identity provider partner worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x identity provider partner worksheet" on page 193
  - "SAML 2.0 identity provider partner worksheet" on page 203

When you have gathered the partner's configuration information, you can then use the Partner wizard in the console to add the partner's federation properties. See "Adding your partner" on page 209.

SAML 1.x service provider partner worksheet

If you are an identity provider using SAML 1.x, you will need to add a service provider partner to your federation. Some information could be supplied to you in a metadata file or all of the information could be supplied to you manually.

Use the following worksheet to gather the necessary information from your partner. You might want to modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 62. Metadata options for adding service provider partner in SAML 1.x federation

Metadata Options	Description	Your values
Enter SAML settings manually **Import metadata file**	Specifies how you will enter data about the partner. You can receive a metadata file from your partner or enter the partner's information manually. If you choose to import a metadata file, you need the file name and its location.	Choose either: • Enter SAML settings manually • Import metadata file and specify file name and path:

Table 63. Contact information for service provider partner in SAML 1.x federation

Contact Information	Description	Your value
Note: This panel is displayed only if you are entering the partner information manually.		
Identity Provider Company Name, URL, and contact person information	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 64. SAML message settings for service provider partner in SAML 1.x federation

SAML Message Settings	Description	Your value
Note: This panel is displayed only if you are entering the partner information manually.		
Provider ID	The URL for the point of contact server of the service provider, which is used as the Provider ID.	Provider ID:
Assertion Consumer Service URL	The URL for the assertion consumer service endpoint at the service provider site.	Assertion Consumer Service URL:
Partner uses Browser POST profile for Single Sign-On	A check box that indicates that the service provider partner uses Browser POST.	One of the following: • Partner uses Browser POST (Select check box.) • Partner does not use Browser POST (Clear check box.)

Table 65. Signature validation information for service provider partner in SAML 1.x federation

Signatures	Description	Your value
Validate Signatures on Artifact Requests	You have the option of validating the SAML message signatures when browser artifact is used. To use this option, select the Validate Signatures check box.	One of the following: • Validate signatures for artifact. (Select check box.) • Do not validate signatures for artifact. (Clear check box.)

Table 65. Signature validation information for service provider partner in SAML 1.x federation (continued)

Signatures	Description	Your value
Select Validation truststore or key - Truststore in Tivoli Federated Identity Manager key service, where the key is stored - Password for the truststore - Public key you will use for validation	If you select to validate messages when browser artifact is used, you must provide a key for the validation. The key must be the public key that corresponds to the private key that your partner uses to sign the messages. **Note:** If you are importing your partner's data, the key is supplied in the metadata file. You will be asked to choose a keystore for the key. Be sure you have created the keystore prior to this task. If you are manually entering your partner's data, be sure you have obtained the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, "Setting up message security," on page 29.	**Metadata method:** - Truststore name: - Truststore password: - Label for key: **Manual method:** - Truststore name: - Truststore password: - Key alias name:

Table 66. Security token settings information for service provider partner in SAML 1.x federation

Configure Security Token	Description	Your value
Sign SAML Assertions	You have the option of signing SAML assertions.	One of the following: - Enable SAML signatures. (Select check box.) - Do not enable signatures. (Clear check box.)
Select Signing Key - Keystore in Tivoli Federated Identity Manager key service, where the key is stored - Password for the keystore - Private key you will use for signing the assertion.	If you choose to sign the assertion signatures, you must select a keystore and a key. **Note:** Create the keystore and key prior to this task. See Chapter 6, "Setting up message security," on page 29.	- Keystore name: - Keystore password: - Key alias name:

Table 66. Security token settings information for service provider partner in SAML 1.x federation (continued)

Configure Security Token	Description	Your value
Include the X509 certificate data	If you choose to sign the SAML assertion, specify whether you want the BASE64 encoded certificate data to be included with your signature. The default action is to include the X.509 certificate data (**Yes**). Or, you could also choose to exclude the X.509 certificate data (**No**).	
Include the X509 Subject Issuer Details	If you choose to sign the SAML assertion, specify whether you want the issuer name and the certificate serial number to be included with your signature. The default action is to exclude (**No**) the X.509 subject issuer details . Or, you could choose to include the X.509 subject issuer details (**Yes**).	
Include the X509 Subject Name	If you choose to sign the SAML assertion, specify whether you want the subject name to be included with your signature. The default action is to exclude the X.509 subject name (**No**). Or, you could choose to include the X.509 subject name (**Yes**).	
Include the X509 Subject Key Identifier	If you choose to sign the SAML assertion, specify whether you want the X.509 subject key identifier to be included with your signature. The default action is to exclude the subject key identifier (**No**). Or, you could choose to include the X.509 subject key identifier (**Yes**).	
Include the Public Key	If you choose to sign the SAML assertion, specify whether you want the public key to be included with your signature. The default action is to exclude the public key(**No**). Or, you could choose to include the public key (**Yes**).	

Table 66. Security token settings information for service provider partner in SAML 1.x federation (continued)

Configure Security Token	Description	Your value
Include the InclusiveNamespaces element	If you choose to sign the SAML assertion, you can select to use the InclusiveNamespaces element in the canonicalization of the assertion during signature creation. The default is unchecked.	
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file or by the custom mapping module will be included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:2.0:assertion` , enter that value in the box. Use && to separate multiple attribute types.	

Table 67. Identity mapping information for service provider partner in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • A custom mapping module • An XSL transformation file containing mapping rules • Leave all options blank to use the identity mapping option that is currently defined for the federation.	The type of identity mapping you will use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. Or, use one of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in “Adding your partner” on page 209.

SAML 1.x identity provider partner worksheet

If you are a service provider using SAML 1.x, you will need to add an identity provider partner to your federation. Some information could be supplied to you in a metadata file or all of the information could be supplied to you manually.

Use the following worksheet to gather the necessary information from your partner. You might want to modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 68. Metadata options for adding identity provider partner in SAML 1.x federation

Metadata Options	Description	Your values
Enter SAML settings manually **Import metadata file**	Specifies how you will enter data about the partner. You can receive a metadata file from your partner or enter the partner's information manually. If you choose to import a metadata file, you need the file name and its location.	Choose either: • Enter SAML settings manually • Import metadata file and specify file name and path:

Table 69. Contact information for identity provider partner in SAML 1.x federation

Contact Information	Description	Your value
Note: This panel is displayed only if you are entering the partner information manually.		
Company Name, URL, and contact person information	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 70. SAML message settings for identity provider partner in SAML 1.x federation

SAML Message Settings	Description	Your value
Note: This panel is displayed only if you are entering the partner information manually.		
Provider ID	The URL for the point of contact server of the service provider, which is used as the Provider ID.	Provider ID:
Source ID • Generate Source ID automatically • Enter explicit value for source ID	You have the option of generating a source ID for the partner or providing one.	Source ID:
Endpoints • Intersite Transfer Service URL • Artifact Resolution Service URL	The URLs for the Intersite Transfer Service and Artifact Resolution Service endpoints.	Intersite Transfer Service URL: Artifact Resolution Service URL:

Table 71. Signature validation information for identity provider partner in SAML 1.x federation

Signature Validation	Description	Your value
SAML Messages for Browser POST are signed and must be validated (required) **Validate Signatures on SAML Messages for Artifact Profile** (optional)	• When browser POST is used as the profile, SAML messages must be signed and validated. Therefore, this option is pre-selected and cannot be deselected. • You also have the option of also validating the SAML message signatures when browser artifact is used.	One of the following: • Validate signatures for artifact. (Select check box.) • Do not validate signatures for artifact. (Clear check box.)
Select Validation Truststore or Key • Truststore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the truststore • Public key you will use for validating your partner's signature	Because Browser POST messages must be signed and validated, you are required to specify a key to validate the signature. If you select to also validate messages when browser artifact is used, the same validation key will be used to validate them. The key you use is the public key that corresponds to the private key that your partner uses to sign messages. **Note:** If you are importing your partner's data, the key is supplied in the metadata file. You will be asked to choose a keystore for the key. Be sure you have created the keystore prior to this task. If you are manually entering your partner's data, be sure that you have obtained the key from your partner and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 6, "Setting up message security," on page 29.	**Metadata method:** • Truststore name: • Truststore password: • Label for key: **Manual method:** • Truststore name: • Truststore password: • Key alias name:

Table 72. Server certificate validation for your identity provider partner in a SAML 1.x federation

Server Certificate Validation for SOAP	Description	Your value
Select Server Validation Certifcate	The public key for the certificate that is displayed during SSL communication with your partner. You and your partner should have discussed which certificate to use. You must have already obtained the certificate and keystore for the certificate. See “Retrieving the server certificate from your partner” on page 58.	Truststore name: Truststore password: Certificate name:

Table 73. Client authentication for SOAP for your identity provider partner in a SAML 1.x federation

Client Authentication for SOAP	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate you will present to the identity provider's server. This is the certificate that you and your identity provider partner agreed that you would present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know which type you must use. If it is basic authentication, you will need a user name and password. If it is client certificate authentication, you will need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure you have agreed with your partner where it will come from and obtained it and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See“Obtaining your client certificate” on page 59	One of the following: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 74. Security token settings information for identity provider partner in SAML 1.x federation

Configure Security Token	Description	Your value
Enable Signature Validation	If your partner signs assertions, you can choose to validate those signatures. In some cases, your partner will require you to validate the signatures.	One of the following: • Enable validation signatures. (Select check box.) • Do not validate signatures. (Clear check box.)
Select Validation Key	Specify the type of signature validation to use.	One of the following: • Use XML signature's KeyInfo to find X.509 certificate for signature validation • Use keystore alias to find public key for signature validation. (The default action.) • Specify the Subject DN expression for the allowable X.509 certificates.
Select key and truststore • Truststore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the truststore • Public key you will use for validating the signature	If you choose to validate the assertion signatures or your partner requires signature validation, you must select a keystore and a key. **Note:** The key you use must be the public key that corresponds to the private key that your partner uses to sign the assertions. Obtain this key and create the keystore prior to this task. Chapter 6, "Setting up message security," on page 29.	• Truststore name: • Truststore password: • Key alias name:
Create multiple attribute statements in the Universal User	Select this check box to keep multiple attribute statements in the groups they were received in. This option might be necessary if your custom identity mapping rules are written to operate on one or more specific groups of attribute statements. If this check box is not selected, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document. The default setting of the check box is not selected and this setting is appropriate for most configurations.	

Table 75. Identity mapping information for identity provider partner in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • A custom mapping module • An XSL transformation file containing mapping rules • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping you will use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. Or, select one of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in “Adding your partner” on page 209.

SAML 2.0 service provider partner worksheet

If you are an identity provider using SAML 2.0, you will need to add a service provider partner to your federation.

Use the following worksheet to gather the necessary information from your partner. You might want to modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 76. Federation to which you are adding a service provider partner in a SAML 2.0 federation

Select Federation	Description	Your value
Federation name	The name of the federation to which you are adding the partner.	

Table 77. Metadata file from your service provider partner in a SAML 2.0 federation

Import Metadata	Description	Your value
Metadata file	The name and path of the file you obtained from your partner that contains your partner's configuration information.	

Table 78. Signature validation for your service provider partner in a SAML 2.0 federation

Signature Validation	Description	Your value
Select which incoming SAML messages and assertions require a signature	Buttons that indicate which incoming messages your partner will sign. The default setting is for the typical set of incoming SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of incoming SAML messages and assertions are signed • All incoming SAML messages and assertions are signed. • No incoming SAML messages and assertions are signed.
Keystore	If your partner will sign messages and assertions, the truststore in which you will store the key that your partner provided to validate its signature in messages. You must have already created the keystore for this key. "Preparing the keystores" on page 29.	Truststore name: Truststore password: Key label:

Table 79. Keystore for storing the encryption key from your service provider partner in a SAML 2.0 federation

Encryption	Description	Your value
Keystore	The truststore in which you will store the key to encrypt messages to your partner. This option is displayed because your partner has provided a public key in its metadata for you to use for encryption. You must have already created the keystore for this key. "Preparing the keystores" on page 29.	Truststore name: Truststore password: Key label:

Table 80. Server certificate validation for your service provider partner in a SAML 2.0 federation

SSL Server Authentication for Artifact Resolution	Description	Your value
Select Server Validation Certifcate	The public key for the certificate that is displayed during SSL communication with your partner. You and your partner should have discussed which certificate to use. You must have already obtained the certificate and added it to your truststore. See "Retrieving the server certificate from your partner" on page 58.	Truststore name: Truststore password: Certificate name:

Table 81. Client authentication for your service provider partner in a SAML 2.0 federation

SSL Client Authentication for Artifact Resolution	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate you will present to the identity provider's server. This is the certificate that you and your identity provider partner agreed that you would present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know which type you will use. If it is basic authentication, you will need a user name and password. If it is client certificate authentication, you will need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure you have agreed with your partner where it will come from and obtained it and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See "Obtaining your client certificate" on page 59.	One of the following: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 82. Partner settings for your service provider partner in a SAML 2.0 federation

Partner Settings	Description	Your value
Session Timeout (seconds)	The number of seconds that a session remains valid when there is no activity. Default value: 3600 seconds.	Session timeout:

Table 82. Partner settings for your service provider partner in a SAML 2.0 federation (continued)

Partner Settings	Description	Your value
Logout Request Lifetime (seconds)	Specifies the maximum time, in seconds, that the logout request remains valid. The default value is 120 seconds.	Logout lifetime:

Table 83. SAML Assertion settings for your service provider partner in a SAML 2.0 federation

SAML Assertion Settings	Description	Your value
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file or by the custom mapping module will be included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of the following type, enter that value in the box: `urn:oasis:names:tc:SAML:2.0:assertion` Use && to separate multiple attribute types.	
Encryption options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes	Check boxes that indicate which assertion parts should be encrypted. If you do not make a selection and leave the boxes blank, no assertion parts in your messages will be encrypted.	Leave blank or choose one or more of the following: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes
Encryption algorithm: • AES-128 • AES-256 • AES-192 • Triple DES	The type of encryption algorithm to use for encrypting data to your partner. If you do not select an algorithm, Triple DES will be used.	Choose one of the following, if you chose an encryption option: • AES-128 • AES-256 • AES-192 • Triple DES

Table 84. Attribute query mapping information for your service provider partner

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you create the file before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 85. Identity Mapping options for your service provider partner in a SAML 2.0 federation

Identity Mapping Options	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping you will use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in “Adding your partner” on page 209.

SAML 2.0 identity provider partner worksheet

If you are a service provider using SAML 2.0, you will need to add an identity provider partner to your federation.

Use the following worksheet to gather the necessary information from your partner. You might want to modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 86. Federation to which you are adding an identity provider partner in a SAML 2.0 federation

Select Federation	Description	Your value
Federation name	The name of the federation to which you are adding the partner.	

Table 87. Metadata file from your identity provider partner in a SAML 2.0 federation

Import Metadata	Description	Your value
Metadata file	The name and path of the file you obtained from your partner that contains your partner's configuration information.	

Table 88. Signature validation for your identity provider partner in a SAML 2.0 federation

Signature Validation	Description	Your value
Select which incoming SAML messages and assertions require a signature	Buttons that indicate which incoming messages your partner will sign. The default setting is for the typical set of incoming SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of incoming SAML messages and assertions are signed • All incoming SAML messages and assertions are signed. • No incoming SAML messages and assertions are signed.
Keystore	If your partner will sign messages and assertions, the truststore in which you will store the key that your partner provided to validate its signature in messages. You must have already created the keystore for this key. "Preparing the keystores" on page 29.	Truststore name: Truststore password: Key label:

Table 89. Keystore for storing the encryption key from your identity provider partner in a SAML 2.0 federation

Encryption	Description	Your value
Keystore	The truststore in which you will store the key to encrypt messages to your partner. This option is displayed because your partner has provided a public key in its metadata for you to use for encryption. You must have already obtained the certificate and imported it into a keystore. Chapter 6, "Setting up message security," on page 29	Truststore name: Truststore password: Key label:

Table 90. Server certificate validation for your identity provider partner in a SAML 2.0 federation

SSL Server Authentication for Artifact Resolution	Description	Your value
Select Server Validation Certifcate	The public key for the certificate that is displayed during SSL communication with your partner. You and your partner should have discussed which certificate to use. You must have already obtained the certificate and added it to your truststore. See "Retrieving the server certificate from your partner" on page 58.	Truststore name: Truststore password: Certificate name:

Table 91. Client authentication for your identity provider partner in a SAML 2.0 federation

SSL Client Authentication for Artifact Resolution	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate you will present to the identity provider's server. This is the certificate that you and your identity provider partner agreed that you would present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know which type you will use. If it is basic authentication, you will need a user name and password. If it is client certificate authentication, you will need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure you have agreed with your partner where it will come from and obtained it and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See "Obtaining your client certificate" on page 59.	One of the following: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 92. Partner settings for your identity provider partner in a SAML 2.0 federation

Partner Settings	Description	Your value
Default Post-Authentication Target URL	The location to which the user should be redirected when the service provider does not provide a target URL during the initial request. This URL must be valid but does not have to be active.	

Table 93. SAML Assertion settings for your identity provider partner in a SAML 2.0 federation

SAML Assertion Settings	Description	Your value
Username to be used for anonymous users	A name identifier that allows a user to access a service through an anonymous identity. The user name entered here is one that the service provider will recognize as a one-time name identifier for a legitimate user in the local user registry. This feature allows users to access a resource on the service provider without having to establish a federated identity. This feature is useful in scenarios where the service provider does not need to know the identity of the user account but only needs to know that the identity provider has authenticated (and can vouch for) the user.	
Map unknown name identifiers to the anonymous username	Specifies that the service provider can map an unknown persistent name identifier alias to the anonymous user account. By default, this option is disabled.	

Table 93. SAML Assertion settings for your identity provider partner in a SAML 2.0 federation (continued)

SAML Assertion Settings	Description	Your value
Create multiple attribute statements in the universal user	Select this check box to keep multiple attribute statements in the groups they were received in. This option might be necessary if your custom identity mapping rules are written to operate on one or more specific groups of attribute statements. If this check box is not selected, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document and in the assertion. The default setting of the check box is not selected and this setting is appropriate for most configurations.	
Encryption options: • Encrypt name identifiers	Check box that indicates whether the name identifiers in assertions should be encrypted.	Select or clear the check box.

Table 94. Attribute query information for identity provider partner

Attribute query	Description	Your value
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file or by the custom mapping module are included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of the following type, enter that value in the box: `urn:oasis:names:tc:SAML:2.0:assertion` Use && to separate multiple attribute types.	

Table 94. Attribute query information for identity provider partner (continued)

Attribute query	Description	Your value
Encryption options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes	Check boxes that indicate which assertion parts to encrypt. If you do not make a selection and leave the boxes blank, no assertion parts in your messages are encrypted. **Encrypt all assetion attributes** indicates whether all attributes in the assertion are encrypted. When this is not selected (set to false), you can manage the encryption of specific attributes through an XSLT SAML token mapping rule.	Leave blank or choose one or more of the following: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes
Encryption algorithm: • AES-128 • AES-256 • AES-192 • Triple DES	The type of encryption algorithm to use for encrypting data to your partner. If you do not select an algorithm, Triple DES will be used.	Choose one of the following, if you chose an encryption option: • AES-128 • AES-256 • AES-192 • Triple DES

Table 95. Attribute query mapping information for identity provider partner

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you create the file before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file (path and name) • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 96. Identity Mapping options for your identity provider partner in a SAML 2.0 federation

Identity Mapping Options	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping you will use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether you will use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in “Adding your partner.”

Adding your partner

After you have configured your role in the federation and gathered information about your partner, you will need to add your partner.

Before you begin

Before beginning this procedure, complete the appropriate partner information worksheet.

- If you are the identity provider, you will add a service provider partner. Use the service provider worksheet for the SAML standard you are using in your federation:
  - “SAML 1.x service provider partner worksheet” on page 188
  - “SAML 2.0 service provider partner worksheet” on page 198

- If you are the service provider, you will add an identity provider partner. Use the identity provider worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x identity provider partner worksheet" on page 193
  - "SAML 2.0 identity provider partner worksheet" on page 203

About this task

After completing the appropriate partner worksheet, use the Partner wizard in the console to add the partner. For descriptions of the fields that you are prompted for by the wizard, refer to the worksheet and the online help.

Note: During the configuration, you might be prompted to restart WebSphere Application Server. Make sure the server has restarted completely before continuing with the task.

Procedure

1. Make sure you have gathered the partner information as described in the worksheets. For example, if you are using a metadata file from your partner, copy the file to an easily accessible location on your computer.
2. Log in to the console. Click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**.
3. The Federations panel is displayed. Select the federation to which you will add the partner and then click **Add partner**. Depending on the SAML standard you are using in the federation, one of the following panels is displayed:

   Metadata Options

   This panel is displayed if you are adding a partner to a SAML 1.x federation. From this panel, click either:
   - **Import Metadata**
   - **Enter SAML data manually**

   Import Metadata

   This panel is displayed if you are adding a partner to a SAML 2.0 federation.
4. Use your completed worksheet as a guide for completing the fields that are displayed in each panel. If you need to go back to a previous panel, click **Back**. If you want to end the configuration, click **Cancel**. Otherwise, click **Next** after you complete each panel.
5. Verify that the settings are correct and click **Finish**. The Add Partner Complete panel is displayed. The partner has been added to the federation, but is disabled by default as a security precaution. You must enable the partner.
6. Click **Enable Partner** to activate this partner.

What to do next

If you have not already provided your configuration information to your partner, you can do that now using the instructions in "Providing federation properties to your partner."

Providing federation properties to your partner

When your partner wants to add you as a partner to their federation configuration, you must provide your partner with the necessary information.

The steps to take are specific to whether you can provide a metadata file or whether you must manually provide the information.

- **Metadata file method**

  If your partner has a way to import your data, you can use the metadata file method whether you have configured a SAML 1.x or SAML 2.0 federation.

  1. Use the console to generate a metadata file that contains the necessary federation configuration and a key for validating response message signatures, if you require validation of the signatures. Follow the instructions in "Exporting federation properties."
  2. You might also need to provide your partner with the appropriate keys and certificates for your role and SAML standard in the federation. See Chapter 6, "Setting up message security," on page 29.

- **Manual method**

  You have the option of manually collecting the necessary configuration instead of exporting the properties to a file, if you configured a SAML 1.x federation.

  Note: Use of a metadata file is preferable because it eliminates the chance of errors being made during the manual input of data.

  If you need to collect information manually, complete the following task.

  1. Use the Federation Properties in the console to obtain the properties. To display the Federation Properties panel for your federation, follow the instructions in "Viewing federation properties" on page 212.

     Use the contents of the Federation Properties panel to guide you to the properties that apply to your federation.
  2. You might also need to provide your partner with appropriate keys and certificates for your federation. See Chapter 6, "Setting up message security," on page 29.

Exporting federation properties

When you want to join a partner's federation, you must supply your federation configuration properties. You can export your federation properties to a file to share them with your partner.

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**.
2. The Federations panel is displayed. Select a federation from the table.
3. Click **Export**. The browser displays a message window that prompts you to save the file containing the exported data. Click **OK**. The browser download window prompts for a location to save the file.
4. Select a directory and metadata file and click **Save**. Metadata file names have the following syntax:

   `federationname_companyname_``metadata.xml`

   For example, for a federation named `federation1` and a company named `ABC`, the metadata file would be named:

   ```
   federation1_ABC_metadata.xml
   ```

   Place the file in an easily accessible location. You will need to provide this file to your partner, when your partner wants to import configuration information for the federation.

Viewing federation properties

Use the Federations properties selection to view the details about an existing federation or to modify an existing federation. This task can be helpful if you need to manually collect your federation properties to share them with your partner.

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**.
2. The Federations panel displays a list of configured federations. Select a federation.
3. Click **Properties** to view properties for an existing federation.
4. Select the properties to modify. Federation properties are described in the online help.
5. When finished viewing or modifying properties, click **OK** to close the Federation Properties panel.

Synchronizing system clocks in the federation

Because security tokens have expiration times, you and your partner's system clocks must be synchronized.

About this task

In your environment, you will need to ensure that the clock on the system where you have the Tivoli Federated Identity Manager runtime and management services component installed is synchronized with your partner.

Refer to the information of your operating system documentation for information about your system clock and time synchronization. Consider using the NTP time synchronization protocol.

Chapter 18. Planning an Information Card federation

The Information Card system allows users to manage their digital identities from various identity providers and use them to access various services that accept these digital identities.

This planning guide reviews the Tivoli Federated Identity Manager implementation of the Information Card standard, and describes how to plan the configuration process. This guide does not provide a comprehensive review of the Information Card standard.

Administrators who are not familiar with the standard should review the Information Card information on the Microsoft web site.

The Tivoli Federated Identity Manager support for Information Card includes deployment of Tivoli Federated Identity Manager in both of the Information Card roles: Managed Identity Provider and Relying Party.

The protocol flow when the user provides an information card in order to authenticate at a Web site resembles the flow for forms-based login, but requires extra steps.

1. User directs the browser to a protected Web page that requires authentication.
2. The site redirects the browser to a login page. In an Information Card-enabled browser, the login page contains an HTML tag that allows the user choose an Information Card to authenticate to the site. When the user selects the tag, the browser invokes an *identity selector*.

   Note: An *identity selector* is a browser plug-in that enables the browser to use the Information Card protocol. The plug-ins are sometimes called *identity agents*.
3. The browser support code for Information Cards invokes the identity selector, and passes it the parameter values supplied by the Information Card HTML tag obtained from the Web site in Step 2.

   The user then selects an Information Card, which represents a digital identity that can be used to authenticate at the site.
4. The identity selector sends the information card to the Tivoli Federated Identity Manager identity provider. The identity provider uses the Tivoli Federated Identity Manager security token service to process the WS-Trust message and WS-Metadata Exchange and then generate a token that contains the user credentials. The identity provider returns the token to the browser.
5. The browser forwards the user credentials to the Web site that protects the requested resource. The site validates the credentials and redirects the browser back to the requested page.

In the protocol flow, the Relying Party and the identity provider do not communicate directly with each other. By default, neither party is aware of the other. The Relying Party does not know which identity provider was selected by the user until the token is received in Step 5. At that time, the Relying Party can learn the identity by examining the Issuer field in the token.

It is possible in Information Card for the identity provider to require identification of the Relying Party, but this is not a requirement, and is typically discouraged.

Overview of the Information Card identity provider

The identity provider supports:

- Issuing managed cards

  The issuing of managed cards is triggered when a user authenticates to a Tivoli Federated Identity Manager identity provider and accesses a card download URL. The URL sends the user a template HTML form, requesting user information that is required in order to issue the card. When the user supplies the necessary information, Tivoli Federated Identity Manager issues the card and sends it to the user's browser. The user can save this card for future use.
- Retrieval of security tokens for managed cards

  This support is provided through the Security Token Service (STS). This component supports two types of SOAP messages from an Information Card identity selector. The SOAP messages are required for an identity selector to obtain a security token for a user's managed information card.

  Note: An *identity selector* is a browser plug-in. It is sometimes called an *identity agent*.

  Only SAML 1.1 security tokens are supported.

Tivoli Federated Identity Manager, when operating as an identity provider, supports the issuing of managed cards, and issues security tokens for managed cards. The support includes the following features:

- Issuing of managed cards
- Endpoints for metadata exchange, and processing of WS-Trust messages
- Support for Information Card claims
- A unique federation to contain the identity provider endpoints
- A trust service chain to convert user identity information into a SAML 1.1 token

Note: Information Card federations do not maintain configuration settings as metadata. This means that for Information Card deployments there is no metadata to export or import between identity providers and relying parties.

Issuing of managed cards

The Tivoli Federated Identity Manager provides support for identity providers to issue managed cards, and to retrieve security tokens from managed cards that have been issued by other authorities.

Tivoli Federated Identity Manager provides a protected endpoint that permits the downloading of a managed card. When a user, through a browser, accesses the endpoint, an HTML template file is loaded and returned the user. The user is prompted to supply information that is required in order to issue the managed card.

The required information is:

- User name

  This is an arbitrary value that the user assigns to the card.
- The set of claims that the card supports.

  A claim is a Uniform Resource Indicator (URI) that represents qualified attribute names. Tivoli Federated Identity Manager uses the list of claims to determine which information (each claim, and its corresponding value) to place into the security token that is generated, at runtime, when the managed card is processed

- When the federation uses an authentication method called *self-issued credential* (or *self-signed SAML assertion*), the user is prompted to post a token generated by a self-issued card as part of the request.

  When the federation uses an authentication method called *username token*, the user does not need to provide this parameter.

Tivoli Federated Identity Manager provides two template HTML pages.

- When the authentication method is username token, the template getcard_ut.html is used.
- When the authentication method is self-issued credential, the template getcard_sss.html is used.

Administrators can modify the template HTML files to best suit the local deployment.

The getcard_* template files contain the following macros, which are replaced with values specific to the request from the end user.

@FORMACTION@
: This macro is replaced with the required form action URL, to which the HTML form is posted.

@USERNAME@
: This macro is replaced with the user name, as supplied by either the login name for the Tivoli Access Manager user or by an authenticated WebSphere user. The Tivoli Access Manager user name is used when WebSEAL is the point of contact server. The WebSphere user name is used when WebSphere is the point of contact server.

  This value can be used pre-populate the card name parameter in the template.

When the user posts the form back to Tivoli Federated Identity Manager, the information is placed into macros in an XML template file called infocard_template.xml. This template file represents the managed card that is returned to the user through the browser.

In most deployments, system administrators will not need to modify the macros in infocard_template.xml. However, the file provides a number of macros that can be modified if needed.

Note: To view a list of macros, see “Replacement macros in the infocard_template XML file” on page 243

Tivoli Federated Identity Manager support for Information Card includes the SAML 1.1 token type only. The SAML 1.1 token type has two representations:

SAML 1.1
: urn:oasis:names:tc:SAML:1.0:assertion

SAML 1.1
: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1

Most managed information cards support both representations. The end user does not select the token type. The @SUPPORTED_TOKENS@ macro in infocard_template.xml is defined to the two SAML representations shown above.

Tivoli Federated Identity Manager supports two methods for the identity selector to authenticate the user to the identity provider security token service. Each methods supports a different replacement template for the @USERCRED@ macro in the information card template (infocard_template.xml).

The type of authentication is specified by the administrator when the federation is configured. The configuration values for the `authenticationMethod` parameter map to template files as follows:

UsernameToken

Maps to the template file **infocard_usercred_usernametoken.xml**

The template file has one replacement macro:

@USERNAME@

This macro is replaced with the user name, as supplied by either the login name for the Tivoli Access Manager user or by an authenticated WebSphere user. The Tivoli Access Manager user name is used when WebSEAL is the point of contact server. The WebSphere user name is used when WebSphere is the point of contact server.

SelfSignedSAML

Maps to the template file **infocard_usercred_selfsignedsaml.xml**.

The template file has one replacement macro:

@PPID@

This macro is replaced with the PPID of the self-issued card that is posted as part of the getcard_sss.html form. This occurs when the federation uses the SelfSignedSAML authentication method.

Tivoli Federated Identity Manager stores this as an alias for the current user in the Tivoli Federated Identity Manager alias service. The alias is used to map the self-issued card back to the Tivoli Federated Identity Manager user when the self-issued card is used (at runtime) to generate a SAML assertion to authenticate to the identity provider security token service.

Identity provider federations

The configuration of federations for Information Card differs significantly from the configuration for federations of other single sign-on protocols, such as SAML 2.0, Liberty, WS-Federation, or OpenID. A primary difference is that the Information Card identity provider has no need to know the relying party that receives the security token. The identity provider security token service interacts only with the identity selector. This eliminates the need to configure any properties that contain information about partners

The concept of partner configuration exists in Information Card configurations only as part of the configuration of token modules used by the trust service.

The key properties that define a federation for an identity provider are:

ProtocolID

Tivoli Federated Identity Manager uses a ProtocolID as a unique identifier. The Information Card federation has the following protocolId syntax:

```
https://<hostname:port>/FIM/sps/<federation_name>/infocard
```

For example:

```
https://www.exampleidentitydemo.com/FIM/sps/csip/infocard
```

Endpoint for obtaining a managed card

An endpoint for processing HTML interaction with an authenticated user, in order to build and download a managed card.

The URL for the endpoint is based on the ProtocolID. For example:

```
https://www.exampleidentitydemo.com/FIM/sps/csip/infocard/getcard.crd
```

The Tivoli Federated Identity Manager component (single sign-on protocol service delegate) for the endpoint completes the following tasks:

1. Prompts the user for information required to generate an information card. The information card information includes the card name and the supported claims. When the authentication method is self-issued credential, a personal information card is generated.
2. When the authentication mechanism is self-issued card (also called SelfSignedSAML), the delegate create and store an alias in the alias service. The alias maps the personal card presented by the user during this process to the user account of the person currently authenticated in the browser session.
3. Generates the managed card from an XML template, with various pieces populated dynamically. The delegate signs the card with the private key of the SSL certificate associated with the point-of-contact server, and then sends the card back to the browser.

Endpoint for exchanging metadata

An endpoint is used by the identity selector (at runtime) to exchange metadata, in order to determine the connection and message formatting requirements (for RST) of the identity provider security token service.

The URL for the metadata endpoint is based on the ProtocolID. For example:

```
https://www.exampleidentitydemo.com/FIM/sps/csip/infocard/mex
```

The metadata exchange endpoint has a template XML file called metadata_template.xml. This file has some macros available for replacement.

Note: Administrators should be able to use the default macros. You do not need to modify the macros in order to use the template file.

The replacement macros for metadata_template.xml are:

@IPSTS@

The URL of the identity provider security token service endpoint for the federation.

@IPPOLICY@

This value consists of WS-Policy information. The information is dependent upon the type of authentication token that is used to authenticate to the identity provider security token service. The WS-Policy information is read from a template file.

@IPCERTIFICATE@
The base-64 encoded public SSL certificate for the point of contact server

Each of the supported authentication methods supports a different replacement template for the @IPPOLICY@ macro in the metadata exchange template.

The template files for each authentication method are:

UsernameToken authentication
metadata_policy_usernametoken.xml

SelfSignedSAML authentication
metadata_policy_selfsignedsaml.xml

The metadata_policy_usernametoken.xml and metadata_policy_selfsignedsaml.xml have no replacement macros. The template files consist of different sets of policy, as appropriate for each method. Information Card administrators do not have to modify these files.

An endpoint for receiving WS-Trust messages
The identity provider security token service has an endpoint that receives WS-Trust messages from the identity selector. The Information Card identity provider module processes the incoming request, including making modifications required for the Tivoli Federated Identity Manager trust service, and communicates with the trust service, to obtain the necessary token.

Information Card claims

Information Card uses information called *claims* to define attributes that can be required in order to fulfill a user request. An information card contains the Uniform Resource Indicators (URIs) for the set of claims that are supported by its issuer.

An identity selector can use the claims information to determine if an identity card can be useful for signing in to a specific relying party. For example, when a relying party requires the e-mail address claim, and the identity provider associated with a given managed card does not support that claim, the identity provider does not offer the managed card as an option for signing in to that relying party.

The Tivoli Federated Identity Provider managed card provider places no restrictions on the set of claims that can be specified in cards. The templates (getcard_ut.html and getcard_sss.html) contain the full set of the standard supported claims. Administrators can add support for additional claims by modifying the templates.

The Information Card identity agent sends a WS-Trust request to the Tivoli Federated Identity Manager module (delegate) for the single sign-on protocol service. The WS-Trust request contains a claims element (wst:Claims) that contains the set of requested claims.

Figure 17 on page 219 shows some example claims.

```
<wst:Claims>
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
        privatepersonalidentifier"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
</wst:Claims>
```

To view the standard set of claims supported by Microsoft, see: http://schemas.xmlsoap.org/ws/2005/05/identity/claims.xsd.

Figure 17. Example claims from a Information Card identity agent

Information Card error pages

The following Page Identifiers are provided:

/infocard/error_get_card.html
Maps to the following page:

```
/infocard/error_get_card.html
```

Used to display an error in HTML when a user is attempting to download a card.

/infocard/error_get_metadata.html
Maps to the following page:

```
/infocard/error_get_metadata.html
```

Used to display an error in HTML when an identity selector user is attempting to download metadata using HTTP GET (rather than SOAP over HTTP/POST).

Overview of the Information Card relying party

The role of Relying Party is similar to the role of a *service provider*, as supported by Tivoli Federated Identity Manager for other single sign-on protocols. The Relying Party consists of a login service implemented a single sign-on protocol service component (delegate) and a WS-Trust chain. The Tivoli Federated Identity Manager implementation supports:

- Reception of SAML 1.x assertion tokens
- The use of login with both self-issued cards and managed cards issued by other identity providers.

In the Information Card model, the Secure Socket Layer (SSL) public key is used to encrypt the token that is sent to the endpoints for the Relying Party. The SSL key is the key of the SSL session established between the browser and the site presenting the Web page (as specified in the embedded OBJECT tags). This means that Tivoli

Federated Identity Manager needs access to the SSL keys used by Point of Contact server. The administrator must configure access to these keys during Tivoli Federated Identity Manager Information Card configuration.

It is recommended that Web sites use X509v3 certificates with logotypes (also known as Extended Validation certificates) instead of SSL server certificates when providing identification of the enterprise.

The Information Card term **Relying Party** refers to a role that is similar to the **Service Provider** role in other single sign-on protocols that are supported by Tivoli Federated Identity Manager.

As a Relying Party, Tivoli Federated Identity Manager supports both Managed and Self-Issuing Identity Providers.

Tivoli Federated Identity Manager configuration enables administrators to configure support for one or both types of providers.

Prior to completing the Tivoli Federated Identity Manager configuration steps, the relying party administrator can obtain public keys from the identity provider, for use when validating digital signatures on assertions received from that provider.

The Tivoli Federated Identity Manager implementation includes support for:

- User access to the relying party
- Information Card claims
- Federations for processing requests
- Token exchange

User access to a relying party

When a user attempts to access a protected resource at a Web site, and the user has not previously established credentials, a *Point of Contact* Web server typically prompts the user to establish credentials by filling out a login page. The use of Information Card in this scenario is dependent on the prior establishment of the following:

- The user must be using a browser that has been enabled for Information Card. Browsers that support for Information Card have an *identity selector* plug-in installed.
- The login page from the Point of Contact that is protecting the resources at the Web site must be tagged with specific OBJECT tags. The OBJECT tags in the page trigger the browser to start the Information Card interaction.
- The URL that the browser accesses must use the HTTPS protocol.

```
<form method="post" action="/FIM/sps/infocard-fed/infocard/login">
   ...
   <input type="hidden" name="TARGET" value="/TheResource"/>
   <object type="application/x-informationCard" name="xmlToken">
      <param name="requiredClaims"
             value="http://schemas.xmlsoap.org/ws/2005/05/identity/
                   claims/privatepersonalidentifier" />
   </object>
   <input type="submit" value="Login"/>
   ...
</form>
```

Figure 18. Example login format for use by Relying Party

Figure 18 shows sample XML elements in the required login format. The login format requires several important parameters:

Form method action

The value of the `action` parameter must be the URL of the Information Card federation endpoint. The Information Card enabled browser redirects to this endpoint to process the security token received from the Identity Provider.

Note: The administrator specifies this endpoint when configuring Tivoli Federated Identity Manager Information Card.

Input type hidden name

The login form should have a `hidden` element with:

- The `name` parameter set to `TARGET`
- The `value` set to the URL to which the browser is redirected when the login process completes.

There is an alternative way to specify the URL to which the browser is to be redirected. The target can be specified using a query string parameter on the value of the `action` parameter. For example, using the values from Figure 18:

```
action=''FIM/sps/infocard-fed/infocard/login?TARGET=/theResource''
```

When WebSEAL is the Point of Contact server, the `%URL%` macro supported by WebSEAL can be used to specify the target URL.

Object type name

The value of the `name` parameter on the `OBJECT` element must be `xmlToken`.

The browser sends this value to the Relying Party. The Tivoli Federated Identity Manager implementation for Information Card Relying Party uses this parameter to access the security token.

Tivoli Federated Identity Manager as a Relying Party supports the following SAML token types:

- URI supported for all provider types:

  `urn:oasis:names:tc:SAML:1.0:assertion`
- URI supported for self-issuing identity providers only:

  `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`

One or more of these type URIs can be specified in the `tokenType` parameter of the `OBJECT` tag.

Relying party federations

Tivoli Federated Identity Manager establishes and uses federations for Information Card in a manner that is similar to, but not identical to, the federations used for other single sign-on protocols. The differences are:

- The Relying Party interaction is part of the authentication process used by the Point of Contact server (for example, WebSEAL) to grants access to protected resources.

  For other single sign-on protocols, the Point of Contact server presents a login page when a protected resource is accessed, and then authenticates the user and produces credentials for the user. For Information Card, Tivoli Federated Identity Manager as a relying party performs the authentication process and produces the credentials for the user.

  The Relying Party becomes aware that a user sign-on is in progress when a security token (assertion) is received at its message endpoint. The Relying Party must then decide whether to accept or reject the security token.
- Unlike a service provider for other single sign-on protocols, the Relying Party does not send messages to the identity provider. The messages are sent by the *Identity Selector*, without the knowledge of the Relying Party
- In a Information Card federation, the identity providers are a set of loosely-federated entities from which the Web site accepts assertion tokens.
- Information Card supports the Self-Issuing Identity provider.

Information Card requires creation of a federation to represent the Relying Party *self*. The term *self* should not to be confused with Self-Issuing Identity provider. The term is used to distinguish the federation originator (creator) from any partners that are subsequently added to the federation. The self entity properties include:

- The login endpoint
- Parameters that indicate the types of tokens that are accepted
- The keystore alias for the private key from the Point of Contact server, for use in Secure Socket Layer (SSL) connections.
- A default mapping rule. The mapping rule can be overridden by a partner's configuration.

The Information Card federation uses the standard Tivoli Federated Identity Manager naming convention for **protocolID**. The syntax is:

```
https://<hostname:port>/FIM/sps/<federation name>/infocard
```

For example, when the host for the federation endpoints is `rp.example.com`, listening on port `443`, with a federation named `MyInfoCard-rp`, the protocolID is:

```
https://rp.example.com:443/FIM/sps/MyInfoCard-rp/infocard
```

Partner federations are needed to represent identity providers. There can be only one self-issuing token partner. There can be any number of Managed Identity provider partners. An **any** Identity Provider partner may also be added. This partner can be used for guest account access.

Self-Issuing partner

Tivoli Federated Identity Manager configures a partner with the protocolId set to:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
```

This partner is used to process Self-issued cards.

Named Managed Identity Provider Partner
A managed provider partner must have a unique `Issuer URI`. The `Issuer` field of the trust chain mapping is set to the `protocolID` value. When assertions from this provider are signed, a public key alias must be configured for the partner. The administrator must import the public key into a Tivoli Federated Identity Manager keystore before configuring the federation. The Tivoli Federated Identity Manager key service should be used to import the key.

Any Provider Partner
The `Any` provider allows the configuration of a wildcard Assertions from these providers must use *one* of the following values for`<saml:SubjectConfirmationMethod>`:

```
urn:oasis:names:tc:SAML:1.0:cm:bearer
urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
```

When the assertion is signed, the assertion must include a `<ds:KeyInfo>` element in the signature containing a public key that is to be used for validating the signatures.

Note: This configuration should only be used for guest user access. In this configuration, all users are mapped to a guest account.

Web site enablement for Information Card

The Tivoli Federated Identity Manager implementation of the Information Card profile interoperates with the Microsoft CardSpace™ Version 1.0 implementation. Both implementations are based on Information Card Profile Version 1.0. This version is supported in Microsoft Internet Explorer Version 7.

Browsers that support Information Card must recognize special HTML or XHTML tags for invoking the Identity Selector, pass encoded parameters on to the Identity Selector on the platform, and POST back the token resulting from the authentication type selected by the user for choice of a digital identity.

Web sites that employ Information Card-based authentication must support two pieces of functionality:

- Addition of HTML or XHTML tags to their login page to request an Information Card-based login
- Code to log the user into the site, using the credentials supplied by the user in the HTTP POST operation

In response to the Information Card-based login, the web site typically responds by:

- Writing the same client-side browser cookie as it would when logins occur based on username-password authentication (or other mechanisms)
- Issuing the same browser redirects

Changes to login pages

HTML extensions such as the OBJECT tag are used to signal to the browser when to invoke the Identity Selector. However, not all HTML extensions are supported by all browsers.

Also, some commonly supported HTML extensions are disabled in browser high security configurations. For example, the OBJECT tag is disabled by high security settings on some browsers, including Internet Explorer.

An alternative to the use of HTML extensions is the use of an XHTML syntax that is not disabled by changing browser security settings. However, not all browsers provide full support for XHTML.

To provide a solution that addresses the range of scenarios, there are two HTML extension formats. Browsers may support one or both extension formats.

OBJECT syntax

Figure 19shows an example of a page that uses the OBJECT syntax to request that the user log in using an Information Card.

```
<html>
<head>
<title>Welcome to Fabrikam</title>
</head>
<body>
<img src='fabrikam.jpg'/>
<form name="ctl00" id="ctl00" method="post"
action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
<center>
<img src='infocard.bmp' onClick='ctl00.submit()'/>
<input type="submit" name="InfoCardSignin" value="Log in"
id="InfoCardSignin" />
</center>
<OBJECT type="application/x-informationCard" name="xmlToken">
<PARAM Name="tokenType"
Value="urn:oasis:names:tc:SAML:1.0:assertion">
<PARAM Name="issuer" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self">
<PARAM Name="requiredClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname">
</OBJECT>
</form>
</body>
</html>
```

Figure 19. OBJECT syntax example

Notice the OBJECT of type `application/x-informationCard`. When the user selects a card, the resulting security token is included in the response (POST) as the `xmlToken` value of the form. Parameters of the Information Card OBJECT are used to encode the required WSSecurityPolicy information in HTML.

In this example, the relying party is requesting a SAML 1.0 token from a self-issued identity provider, supplying the required claims `emailaddress`, `givenname`, and `surname`.

Note: You can omit the Issuer to indicate that *any* issuer of an Information Card available in the browser for the user is acceptable.

XHTML syntax

The XHTML syntax is as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:ic>
<head>
<title>Welcome to Fabrikam</title>
</head>
<body>
<img src='fabrikam.jpg'/>
<form name="ctl00" id="ctl00" method="post"
action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
<ic:informationCard name='xmlToken'
style='behavior:url(#default#informationCard)'
issuer="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self"
tokenType="urn:oasis:names:tc:SAML:1.0:assertion">
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
optional="false" />
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
optional="false" />
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
optional="false" />
</ic:informationCard>
<center>
<input type="submit" name="InfoCardSignin" value="Log in"
id="InfoCardSignin" />
</center>
</form>
</body>
</html>
```

Figure 20. Example of InfoCard XHTML syntax

Identity selector invocation parameters

The parameters to the OBJECT and XHTML Information Card objects are used to encode information in HTML. In cases where an Identity Selector is used in a Web services context, this information would be supplied as WS-WSSecurityPolicy information through use of WSMetadataExchange.

The following list shows parameters supported by the Information Card standard for Identity Selector invocation.

Note: All parameters are optional. None of them are required.

issuer This parameter specifies the URL of the security token service (STS) from which to obtain a token. When omitted, no specific STS is requested. The special value `http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self` specifies that the token should come from a self-issued identity provider.

Note: This parameter is not supported by Tivoli Federated Identity Manager

issuerPolicy
This parameter specifies the URL of an endpoint from which the WS-SecurityPolicy can be retrieved using WS-MetadataExchange. If omitted, the value `<issuer>/mex` is used. This endpoint must use HTTPS.

Note: This parameter is not supported by Tivoli Federated Identity Manager

tokenType
This parameter specifies the type of the token to be requested from the STS as a URI. This parameter can be omitted when the STS and the Web site point of contact have either previously agreed what token type is to be provided or if the Web site is willing to accept *any* token type.

requiredClaims
This parameter specifies the types of claims that must be supplied by the identity. If omitted, there are no required claims. The value of `requiredClaims` is a space-separated list of URIs, each specifying a required claim type.

optionalClaims
This parameter specifies the types of optional claims that may be supplied by the identity. If omitted, there are no optional claims. The value of `optionalClaims` is a space-separated list of URIs, each specifying a claim type that can be optionally submitted.

privacyURL
This parameter specifies the URL of the human-readable privacy policy of the site, if provided.

privacyVersion
This parameter specifies the privacy policy version. This must be a value greater than `0` if a `privacyUrl` is specified. If this value changes, the UI notifies the user and allows them to review the change to the privacy policy.

Example of a WebSEAL login page

This is an example of the WebSEAL login.html that has been modified with the OBJECT tags shown highlighted in **`bold font`**.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<!-- Copyright (C) 2000 Tivoli Systems, Inc. -->
<!-- Copyright (C) 1999 IBM Corporation -->
<!-- Copyright (C) 1998 Dascom, Inc. -->
<!-- All Rights Reserved. -->
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>Access Manager for e-business Login</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<B>Access Manager for e-business Login (www-default---2)</B>
<BR>
%ERROR%
<BR><BR>
<!--- DO NOT TRANSLATE OR MODIFY any part of the hidden parameter(s) --->

<!---
  The following block of code provides users with a warning message
  if they do not have cookies configured on their browsers.
  If this environment does not use cookies to maintain login sessions,
  simply remove or comment out the block below.
--->

<!--- BEGIN Cookie check block --->
<!---
<! ..... edited from this example for brevity ....
<!--- END Cookie check block --->

<BR>
   <form name="ctl00" id="ctl00" method="post"
        action="https://example.com:443/FIM/sps/infocard/login">
      <center>
          <input type="submit" name="InfoCardSignin" value="Log in"
          id="InfoCardSignin" />
      </center>
      <OBJECT type="application/x-informationCard" id="oCard" name="xmlToken">
        <PARAM Name="tokenType" Value="urn:oasis:names:tc:SAML:1.0:assertion">
        <PARAM Name="issuer" Value=
            "http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self">
        <PARAM Name="requiredClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/
   claims/privatepersonalidentifier">
        <PARAM Name="optionalClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
">      </OBJECT>
    </form>
   </BODY>
</HTML>
```

Figure 21. Example WebSEAL login page with OBJECT tags

Configuration requirements for Information Card

Requirement for WebSphere Version 6.1

Tivoli Federated Identity Manager supports Information Card on WebSphere Application Server 6.1. Information Card is not supported on WebSphere 6.0.

Information Card uses the encryption algorithm **rsa-oaep-mgf1p** for key wrap. This algorithm is supported by WebSphere 6.1, but is not available on WebSphere 6.0.

Tivoli Federated Identity Manager requires application of a fix pack for WebSphere Application Server 6.1. See the hardware and software requirements on the Tivoli Federated Identity Manager information center for the required fix pack level.

Updating the cryptography policy for Information Card

The encryption algorithms used by Information Card require strong cryptographic library support. This mean that a replacement is needed for the default Java security files local_policy.jar and US_export_policy.jar.

About this task

Use of encryption technology is controlled by United States law. IBM Java Solution Developer Kits (SDKs) include strong but limited jurisdiction policy files. To deploy Information Card with Tivoli Federated Identity Manager, you must first obtain the unlimited jurisdiction Java Cryptography Extension (JCE) policy files.

To review the security information for IBM Java SDKs, access the following URL:

```
http://www.ibm.com/developerworks/java/jdk/security/index.html
```

To obtain the unlimited jurisdiction policy files:

Procedure

1. Update WebSphere with unrestricted Java Cryptography Extension (JCE) policy files. Access: http://www.ibm.com/developerworks/java/jdk/security/index.html
2. Select the link to the SDK that matches your environment, for example, for Java 1.5, the SDK is J2SE 5.0. You will see a page that displays the heading Security Information.
3. Select the link: **IBM SDK Policy Files**.

   Note: After you click this link, you will be redirected to the policy file in the SDK that is compatible with your version of Java; note, however, that the version number of the SDK might not be the same as the version number of the Java version you are using. For example, for Java 1.5 you might be directed to the SDK 1.4.
4. You will be prompted to log in using your IBM user ID and password. If you do not have an IBM user ID and password, you will need to register. Follow the registration link on the login page.
5. Log in.
6. When prompted, select the .zip file for the version of Java you are using. Then click **Continue** to begin the download.
7. Unpack the .zip file. The JAR files are:
   - local_policy.jar
   - US_export_policy.jar
8. Place the files in the following directory:

   ```
   your_Java_runtime_installation_dir/jre/lib/security
   ```

   For example, your Java runtime might have been installed as part of the embedded version of WebSphere Application Server. In this case, the directory might be

```
/opt/IBM/FIM/ewas/java/jre/lib/security
```

Information Card requirement for alias service

The alias service must be configured if managed cards backed with self-issued-credential authentication are to be used.

Decryption key from point of contact server

Information Card configuration requires the specification of a key for decrypting messages in the federation. Decryption is required.

This means that a decryption key alias must be added to the Tivoli Federated Identity Manager keystore. The key must be the point of contact server's private key. The key must be imported using the Tivoli Federated Identity Manager key service.

This means that a decryption key alias must be added to the Tivoli Federated Identity Manager keystore. The key is from whichever Web site presents, to the Information Card-enabled browser, the HTML page tagged with the required OBJECT tags. The site can be the point of contact server, but does not have to be. The URL that results in the tagged page *must* use SSL. For example:

```
https://pointofcontact.example.com/FIM
```

The SSL key used for the URL must be imported into a Tivoli Federated Identity Manager keystore for the Relying Party.

Note: When the SSL key is changed or updated for the Web site or point of contact, the administrator must also update the Tivoli Federated Identity Manager keystore with the new SSL key. This may also include modification of the configuration to update the keystore alias.

Information Card time synchronization requirements

Successful deployment of Information Card is dependent on time synchronization between systems. The following requirements must be met:

- When the UsernameToken method of authentication is configured for a federation, then time must be synchronized between the identity provider and the relying party systems.
- When the self-issued credentials method of authentication for a managed card is used, the browser system (which hosts the browser with Information Card functionality) must also be time synchronized.
- When a self-issued card is used to login to the relying party, the browser system (which hosts the browser with Information Card functionality) must also be time synchronized.

The required time synchronization can be specified by the **clock skew** property for each Information Card federation. You can use the Tivoli Federated Identity Manager administration console to modify this property from the federation partner properties panel.

Identity mapping for Information Card

Identity provider

The Tivoli Federated Identity Manager support for Information Card identity providers uses a trust chain that contains modules to perform the standard actions of validate, map, and issue.

The validate operation is performed on the authentication token sent by the identity selector to represent the user. The token is either a Username token or SAML assertion. The SAML assertion is used with self-issued credentials authentication.

The mapping module can be one of the following:
- XSLT mapping module
- Tivoli Directory Integrator module
- A custom-developed Java map module

Tivoli Directory Integrator is commonly used as the mapping module with Information Card. In Information Card deployments, a primary goal of the trust chain is to identify claims values and populate them in the security token service universal user. The claims values can come from external data sources, such as an LDAP registry. The Tivoli Directory Integrator module can, for example, convert LDAP entries for a user into the corresponding claims values, as defined in the schema that Microsoft has specified.

Tivoli Directory Integrator modules can also easily be used to combine claims values from a variety of sources. For example, some claims values might come from an LDAP registry, while others would originate with other sources, such as databases, Java or JavaScript code, or other web services.

The output of the mapping module is used to produce a SAML 1.1 token in *issue* mode.

Relying party

The trust chain for a relying party federation consists of:
- A SAML 1.1 token module, in validate mode.
- The default Map module
- The IVCred token module, in issue mode.

The federation wizard prompts the administrator to specify identity mapping rules, using XSLT, as appropriate for the deployment. The mapping rules use the attributes of the assertions or the information in the claims to determine the use identity.

The SAML token modules create STSUniversalUser attributes for each attribute in the SAML assertion. The name, namespace and value for each SAML attribute are used to set the STSUniversalUser/Attribute name, type and value.

Identity provider configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of Information Card federations. The wizard prompts you to supply properties for your deployment.

This worksheet describes the prompts. Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example:

```
infocard-idp
```

Federation role
Select identity provider.

Company name
The wizard requests contact information. The Company Name field is required. This can be any string. Other fields are optional.

Federation protocol
Select Information Card.

Point of contact server
The server that acts as initial point of contact for incoming requests. For example:

```
https://pointofcontact.example.com/FIM
```

Note: For Information Card support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

SSL Endpoint Key Identifier

The configuration wizard asks you to specify a key to use for decryption operations for the federation. The key must be the key used by the point of contact server for SSL operations.

The wizard asks for this key on the **Infocard Configuration Settings** panel. You specify the key by selecting the Keystore and then the Key.

Note: You must import this key from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federation.

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

Key to select
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Authentication option

You will select one authentication option:

- Authentication with a self-issued card

- Authentication with a username and password

Authentication with a self-issued card is the default option.

The choice of authentication option determines the default value for the property **Download card template file**.

Download card template file

This is an HTML template file that prompts you to enter the input parameters needed to issue a managed Information Card. The configuration wizard provides default values. You can use the defaults unless you have modified and renamed the template files.

- When you select Authentication with a self-issued card, the default value is:

  `/infocard/getcard_sss.html`
- When you select Authentication with a username and password, the default value is:

  `/infocard/getcard_ut.html`

Information card template file

This is an HTML template file that comprises the Information Card that is sent back to you. Default file:

`/infocard/infocard_template.xml`

Information card image file

This is the image file to use for the Information Card. It must be located in the directory for the current locale. The default value is identical for both authentication options. Default file:

`/infocard/fim_infocard.gif`

Card expiration

This property specifies the number of days from the issue date for which the information card is valid. The default value is identical for both authentication options. Default value:

`365`

Identity mapping options

You must select one of the following options:

- Use XSL for identity mapping

  Select this option when you want to use an XSLT mapping rule. You must provide the name of a file that supplies identity mapping rules. Tivoli Federated Identity Manager provides a sample identity mapping rules file for Information Card identity providers federations:

  /*installation_directory*/examples/ip_infocard.xsl
- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your Information Card federation.
- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your Information Card federation.

Table 97. Worksheet for identity provider federation properties

Property	Your value
Federation name	

Table 97. Worksheet for identity provider federation properties (continued)

Property	Your value
Role	Identity Provider
Company Name	
Federation Protocol	Information Card
Point of Contact server	
SSL Endpoint Key Identifier: Keystore	
SSL Endpoint Key Identifier: Keystore password	
SSL Endpoint Key Identifier: Key to select	
Authentication option	
Download card template file	
Information card template file	
Information card image file	Default: /infocard/fim_infocard.gif
Card expiration	Default: 365 days
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Relying party configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of Information Card federations. The wizard prompts you to supply properties for your deployment.

This worksheet describes the prompts. Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example, `infocard-rp`.

Federation role
Select service provider. This value is required for the relying party.

Company name

The wizard requests contact information. The Company Name field is required. This can be any string. Other fields are optional.

Federation protocol

Select Information Card.

Point of contact server

The server that acts as initial point of contact for incoming requests. For example:

```
https://pointofcontact.example.com/FIM
```

Note: For Information Card support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Decryption

The configuration wizard asks you to specify a key to use for decryption operations for the federation. The key must be the key used by the point of contact server for SSL operations.

The wizard for asks for this key on the **Decryption** panel. You specify the key by selecting the Keystore and then the Key.

Note: You must import this key from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federation.

Keystore

The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password

Password required to access the specified keystore.

Key to select

The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Standard Partner

The wizard prompts you to select one option:

- Add a partner that can handle any identity provider

  This option is the default.

  Selecting this option results in a partner being automatically added. This partner configuration can accept any Information Card identity provider, including a self-issuing provider.
- Add a partner that can handle the self-issuing identity provider

  Selecting this option results in a partner being automatically added. This Tivoli Federated Identity Manager partner only accepts personal cards issued by the self-issuing provider built into the browser.
- Do not add a standard partner

  Selecting this option results in no standard partners being added. The administrator must explicitly add partners using the Tivoli Federated Identity Manager console Add Partner wizard.

Identity mapping options

You must select one of the following options:

- Use XSL for identity mapping

  Select this option when you want to use an XSLT mapping rule. You must provide the name of a file that supplies identity mapping rules. Tivoli Federated Identity Manager provides a sample identity mapping rules file for Information Card identity providers federations:

  `/installation_directory/examples/rp_infocard.xsl`
- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your Information Card federation.
- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your Information Card federation.

Table 98. Worksheet for relying party federation properties

Property	**Your value**
Federation name	
Role	Service Provider
Company Name	
Federation Protocol	Information Card
Point of Contact server	
Decryption: Keystore	
Decryption: Keystore password	
Decryption: Key to select	
Standard Partner	Default option: **Add a partner that can handle any identity provider.** Your option:
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Managed partner worksheet

When you create a federation for an identity provider, a partner is automatically created.

After you create a federation for a relying party, you can choose one of several options for configuring a partner. When you choose not to add a standard partner, you can later create a partner for the federation. When you do this, you will need to provide some configuration values.

The Tivoli Federated Identity Manager console provides a wizard to guide you through this process.

Identity Provider Company name
Contact information.

Security Token Issuer
This value is used to set the protocolID and endpoint URL in etc/feds.xml and the Issuer field in the STS chain mapping configuration. For example:

```
https://example.com
```

Maximum allowable clock skew between hosts (seconds)
This is the maximum allowable clock skew between the relying party host and the identity provider host. The clock skew value is used during validation of the assertion's validity period.

The default value is 60 seconds.

Validate signatures on Information Card tokens
You can select this checkbox to specify that incoming security tokens must be signed. When you select this option, you must use the additional configuration properties to specify the public key that is to be used to validate the digital signature.

Type of signature validation key
You must select one of the following:

- Public key from the KeyInfo in the signature of the Information Card token

  You can choose this option if you do not want to distribute and update public keys, and need only to ensure that token integrity is maintained.
- Public key from a keystore

  This public key must have previously been obtained from the managed identity provider and imported into a Tivoli Federated Identity Managerkeystore using the Key Services.

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

Key to select
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Table 99. Worksheet for managed partner configuration properties

Property	Your value
Identity Provider Company name	
Security Token Issuer	

Table 99. Worksheet for managed partner configuration properties (continued)

Property	Your value
Maximum allowable clock skew between hosts (seconds)	
Validate signatures on Information Card tokens	
Type of signature validation key	If validating signatures, select one: • Public key from the KeyInfo in the signature of the Information Card token • Public key from a keystore
Keystore	*When using **Public key from a keystore***:
Keystore password	
Key to select	

Chapter 19. Configuring an Information Card federation

Verifying Information Card dependencies

Before you begin

Before you use the federation creation wizard, ensure that the Information Card dependences have been met.

Procedure

1. Verify that you are installing on WebSphere Application Server 6.1 Older versions are not supported. See "Requirement for WebSphere Version 6.1" on page 227
2. Verify that you have the correct encryption libraries. See "Updating the cryptography policy for Information Card" on page 228.
3. Review whether you need to configure the alias service. See "Information Card requirement for alias service" on page 229
4. Ensure that you have imported the encryption key for the point of contact server This key must be imported into the Tivoli Federated Identity Manager Key service.

Configuring an Infocard federation

Use the federation wizard to create and configure an Infocard federation.

Before you begin

Ensure that you have prepared configuration information before using the wizard to create the federation. The planning activities are described in a series of topics in this guide. See Chapter 2, "Overview of configuration tasks for federated single sign-on," on page 15.

About this task

To use the federation wizard to create and configure an Infocard federation, complete the following steps:

Procedure

1. Log in to the Integrated Solutions Console and click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets are displayed. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The wizard presents a series of configuration panels.
3. Use your completed worksheet to provide values at each panel. Supply the necessary values, and then click **Next** to proceed to the next panel. If you need to go back to adjust a configuration setting, click **Back**. You can view the online help for information about specific fields.
   a. The first series of panels requests settings for the federation name, role, protocol, and point of contact server.

b. Next, the Infocard configuration panel requests the values needed for an Infocard identity provider or relying party.

c. The last series of panels requests settings for the identity mapping configuration.

When you finish entering configuration settings, the Summary panel is displayed.

4. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.

Configuring WebSEAL as a point of contact server for an Information Card federation

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the Information Card federation.

Before you begin

The Create Federation Complete portlet provides a button that you can use to obtain a configuration utility.

About this task

You must obtain the utility and run it. Complete the following steps:

Procedure

1. Click **Download Tivoli Access Manager Configuration Tool**
2. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
3. Return to the management console, and Click **Done** to return to the Federations panel.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation we will complete other tasks first.
4. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -cfgfile webseald-instance_name.conf
    -action tamconfig
   ```

   You will need to know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -cfgfile webseald-default -action tamconfig
```

For more information, see:

- Appendix A, “tfimcfg reference,” on page 491

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log in to the administration console.
2. Click Tivoli Federated Identity Manager > Manage Configuration > Point of Contact
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Specifying a persona index

A persona index is a collection of several sets of attributes, available to a user on an identity provider. The user can specify attributes that describe a persona. For example, a user might have a work persona containing work email address and telephone, and a home persona containing personal email address and telephone. These personas might be called *work* and *home*.

When a user downloads a managed card, the user might want to associate that managed card with a particular persona, so that when a single sign-on token is requested for the card, the identity provider can determine which set of persona attributes to use to populate the token.

The use of personas is enabled by the use of an optional form field parameter called `userdata`. This parameter can be included in getcard_ut.html and getcard_sss.html template pages.

This input field is not included in the shipped template files, but is supported.

When this parameter is provided, a replacement macro called @USERDATA@ can be populated in the infocard_template.html. The @USERDATA@ macro is used in the CardId part of the infocard_template.html file.

The default infocard_template.html file contains the following macro pattern for CardId:

```
<InformationCardReference>
  <CardId>@IPSTS@/@UUID@</CardId>
  <CardVersion>1</CardVersion>
</InformationCardReference>
```

When administrators want to make use of @USERDATA@, a suggested macro pattern is:

```
<InformationCardReference>
  <CardId>@IPSTS@/@UUID@/@USERDATA@</CardId>
  <CardVersion>1</CardVersion>
</InformationCardReference>
```

The CardId information is part of the RST that will be sent by the identity selector to Tivoli Federated Identity Manager when requesting a single sign-on token. That information allows mapping rules to read and differentiate which persona to use.

Using this pattern, a user would be prompted for their persona index when downloading a managed card, and the card would be tied to a particular persona. A user could download different cards for each persona they have specified at the identity provider. The mapping rule in the STS trust chain can read the CardID (and hence the persona index) and populate the runtime identity token with attributes from the correct persona.

Chapter 20. Information Card reference

Replacement macros in the infocard_template XML file

The replacement macros for infocard_template.xml are:

@IPSTS@
The Uniform Resource Locator (URL) of the identity provider endpoint for the federation.

@IPMEX@
The Uniform Resource Locator (URL) of the identity provider Metadata exchange endpoint for the managed card. Note that the URL is specific to the authentication type used.

@UUID@
This macro is replaced with a randomly generated universal user identifier (UUID). This value ensures that the Card identity is unique.

@USERDATA@
This macro is not included in the default file. You can add this macro to the CardId container when you want to specify attributes. This macro is useful when users in your deployment have multiple personas. The users can provide attributes that identify the persona to be used.

@CARDNAME@
The card name that the user specified in the response posting to the form getcard_ut.html or getcard_sss.html.

@CARDIMAGE@
A Multi-purpose Internet Email Extension (MIME) encoded image file that is displayed to the user by the identity selector. There is one image file for each federation.

@ISSUETIME@
The time the card is issued. The time is calculated at runtime.

@EXPIRETIME@
The time the card expires. The time is calculated by adding card the value of the card *lifetime* to the issue time.

@IPCERTIFICATE@
This is the base64-encoded public certificate configured for the federation. It should also be the public certificate of the SSL endpoint for the point of contact server.

@USERCRED@
This is a piece of metadata about the type of credential that is used by the identity selector to authenticate the user to the identity provider (security token service) endpoint. The metatdata comes from another template file, depending on the type of authentication used.

Tivoli Federated Identity Manager support for Information Card includes support for two forms of authentication:

- Username token

  The metadata for the user credential is loaded from the template file inforcard_usercred_usernametoken.xml.

- Self-issued credential

  The metadata for the user credential is loaded from the template file infocard_usercred_selfsignedsaml.xml.

@SUPPORTED_TOKENS@
Tivoli Federated Identity Manager support for Information Card includes the SAML 1.1 token type only. There are two default representations.

@SUPPORTED_CLAIMS@
The set of claims supported by this card. These values come from the form posted by the user in getcard_*.html. The values must be presented in the XML format dictated by the Information Card specifications.

Information Card claims

The following list shows the claim types, with the URI and description for each. The claims types are summarized here for convenience, but users should consult the official list in the referenced schema.

Note: Information Card support in Tivoli Federated Identity Manager is not limited to this set of claims.

First Name
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname

Preferred name or first name of a subject. RFC 2256 uses `givenName` states: "This attribute is used to hold the part of a person's name which is not their surname nor middle name."

Last Name
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname

Surname or family name of a subject. RFC 2256 uses `sn` and states: "This is the X.500 surname attribute which contains the family name of a person."

Email Address
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress

Preferred address for the `To:` field of email to be sent to the subject, usually of the form `<user>@<domain>`.

The term `mail` is used by inetOrgPerson using RFC1274, which states: "This attribute type specifies an electronic mailbox attribute following the syntax specified in RFC 822."

Street Address
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/streetaddress

Street address component of address information for the subject.

RFC 2256 uses the term `street`, and states: "This attribute contains the physical address of the object to which the entry corresponds, such as an address for package delivery." Its content is arbitrary, but typically given as a PO Box number or apartment or house number followed by a street name. For example, 303 Mulberry St.

Locality Name or City
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality

Locality component of the address information for a subject. RFC 2256 uses the term `l`, and states: "This attribute contains the name of a locality, such as a city, county or other geographic region." For example, Austin.

State or Province

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/stateorprovince

Abbreviation for state or province name of the address information for a subject. RFC 2256 uses the term `st`, and states: "This attribute contains the full name of a state or province. The values should be coordinated on a national level and if well-known shortcuts exist, like the two-letter state abbreviations in the US, these abbreviations are preferred over longer full names."

For example, the abbreviation `TX` is used to indicate the state of Texas.

Postal code

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/postalcode

Postal code or zip code component of the address information for a subject. X.500(2001) uses the term `postalCode`, and states: "The postal code attribute type specifies the postal code of the named object. If this attribute value is present, it will be part of the object's postal address - zip code in USA, postal code for other countries."

Country

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country

Country of a subject. RFC 2256 uses the term `c` and states: "This attribute contains a two-letter ISO 3166 country code."

Telephone Number

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/homephone

Primary or home telephone number of a subject. The term `homePhone` is used in inetOrgPerson using RFC 1274, which states: "This attribute type specifies a home telephone number associated with a person."

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Secondary or Work Telephone Number

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/otherphone

Secondary or work telephone number of a subject. X.500(2001) uses the term `telephoneNumber` and states: "This attribute type specifies an office/campus telephone number associated with a person."

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Mobile Telephone Number

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/mobilephone

Mobile telephone number of a subject. The term `mobile` is used by inetOrgPerson using RFC 1274, which states: “This attribute type specifies a mobile telephone number associated with a person.”

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Date of Birth

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth

The date of birth of a subject in a form allowed by the `xs:date` data type.

Gender

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/gender

Gender of a subject. The value must be one of the following string values:

0 Unspecified

1 Male

2 Female

Use of these values allows them to be language neutral.

Private Personal Identifier
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier

A private personal identifier (PPID) that identifies the subject to a relying party. The word **private** means that the subject identifier is specific to a given relying party and therefore is known only to (or *private to*) that relying party. The PPID of a subject at one relying party cannot be correlated with the PPID for the same subject at another relying party.

Web Page
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/webpage

The Web page of a subject expressed as a URL.

Federation properties for identity providers

When you create an Information Card federation for an identity provider, the configuration wizard automatically assigns default values to some properties. You cannot modify these properties during the initial configuration, but you can modify them after the initial configuration completes.

Federation identification

Federation name
An arbitrary string that you choose to name this federation.

For example, for a managed identity provider:

```
infocard-idp
```

Company name
The wizard requests contact information. The only field that is required in the Company name. This can be any string.

Single sign-on properties

Provider ID
A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the identity provider URL. Optionally it can include a port number. For example, for a federation named infocard_fed:

```
https://idp.example.com/sps/infocard_fed/infocard
```

Download Card Endpoint
Endpoint to create and download a managed card. The extension of the file name must be .crd. The default value is

```
Provider_ID/getCard.crd
```

Metadata Exchange Endpoint
The endpoint used by identity selectors to request metadata about the identity provider's Security Token Service (STS). The default value is

```
Provider_ID/mex
```

Security Token Service Endpoint

The endpoint used by identity selectors to request security tokens for a user as part of an Information Card authentication. The default value is

`*Provider_ID*/sts`

Alias Management Endpoint

The endpoint used to manage the association, or link, between a self-issued card and a user's Tivoli Federated Identity Manager account. The link is established when a user downloads a managed card using the "self-issued card" authentication mechanism, and this endpoint may be used to review and delete that linking. The default value is

`*Provider_ID*/alias`

This property is not used when you have selected the authentication option for username and password.

Authentication option

You can change the authentication option to one of the following:

- Authentication with a self-issued card
- Authentication with a username and password

Download card template file

This is an HTML template file that prompts the user to enter the input parameters needed to issue a managed Information Card.

- When you selected Authentication with a self-issued card, the default value was:

  `/infocard/getcard_sss.html`
- When you selected Authentication with a username and password, the default value was:

  `/infocard/getcard_ut.html`

Information card template file

This is an HTML template file that comprises the Information Card that is sent back to you. Default file:

`/infocard/infocard_template.xml`

The default value is the same for both authentication options.

Information card image file

This is the image file to sue for the Information Card. It must be located in the directory for the current locale. The default value is identical for both authentication options. Default file:

`/infocard/fim_infocard.gif`

The default value is the same for both authentication options.

Metadata Card Template

The name of the file to use as the template for the Information Card's metadata. The default file is:

`/infocard/metadata_template.xml.`

Self Signed SAML Credentials Metadata Policy

The name of the policy file to use for the self-signed SAML credentials metadata. The default file is:

`/infocard/metadata_policy_selfsignedsaml.xml`

This field is only displayed when you select **Authenticate with a self-issued card**.

Username Credentials Metadata Policy

The name of the policy file to use for the username credentials metadata. The default file is:

```
/infocard/metadata_policy_usernametoken.xml
```

This field is only displayed when you select **Authenticate with a username and password**.

Card expiration

This property specifies the number of days from the issue date for which the Information Card is valid. The default value is identical for both authentication options. Default value:

```
365
```

SSL Endpoint key identifier

Note: This is the key that you needed to import this key from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federations.

Keystore

The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password

Password required to access the specified keystore.

List Keys

The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use

Information Card signing key identifier

The public and private key pair that is used to sign newly issued Information Cards.

Keystore

The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password

Password required to access the specified keystore.

List Keys

The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use

Token module properties

When the Information Card federation is initially configured, the trust chain is automatically built and configured. The trust chain contains trust modules that require configuration. The properties in this section can be changed.

Enable one-time assertion use enforcement

Use the assertion only one time and do not cache it for future use. This is enabled by default.

This property is used only with self-issued card authentication

Skip password validation

Do not perform password validation for the Username token. The default is unchecked, which means that password validation occurs.

This property is used only with username and password authentication.

Amount of time before the issue date that an assertion is considered valid (seconds)

Default: 60 seconds. There is no minimum or maximum value enforced.

Amount of time the assertion is valid after being issued (seconds)

Default: 60 seconds. There is no minimum or maximum value enforced.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance

This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance

Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties

Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Federation properties for relying party

Federation identification

Federation name

An arbitrary string that you choose to name this federation.

For example, for a relying provider:

```
infocard-rp
```

Company name

The wizard requests contact information. The only field that is required in the Company name. This can be any string.

Single sign-on properties

Provider ID

A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the identity provider URL. Optionally it can include a port number. For example, for a federation named infocard_fed:

```
https://rp.example.com/sps/infocard_fed/infocard
```

Authentication URL
The URL to which the user sends authentication requests. This value cannot be changed on the Properties panel. For example, for a federation named infocard_fed, the authentication URL would be:

```
https://idp.example.com/sps/infocard_fed/infocard/login
```

Decryption key properties

The key to use for decrypting incoming tokens. Note that this must be the same key that is used for SSL by the point of contact server (for example, WebSEAL).

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties
Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Properties for identity provider partners for relying party federations

Federation identification

Member of federation name
The federation to which this partner has been added. You cannot modify this property.

For example, the identity provider is now a partner of the relying provider federation:

```
infocard-rp
```

Partner role
Identity provider. You cannot modify this property.

Status The Partner properties page displays a partner status of Enabled or Disabled. Partners must be enabled (activated) before they can participate in a federation.

- When partner status is Disabled, click Enable to activate the partner.
- When partner status is Enabled, click Disabled to deactivate the partner.

Identity provider company name
The name of the partner company. This can be any string. The space character is allowed. This field is required.

Company URL
The URL of the partner company. This field is optional. For example:

```
http://www.example.com
```

Contact person
Optional contact information for the administrator. You can use the Other information field if necessary.

Token properties

Security Token Issuer
Specify the identity provider's unique issuer Uniform Resource Identifier (URI). This value must be used in the saml:Issuer element of the saml:Assertion. Following is an example:

```
https://example.com
```

You can enter an asterisk (*) to indicate that any identity provider is acceptable.

Maximum allowable clock skew between hosts (seconds)
Specify an integer value indicating the maximum amount of allowable clock skew, in seconds, between the Relying Party host and the Identity Provider host. You must specify a minimum value of zero seconds for this field. The default value is 60. This field is only available when the federation uses the Authenticate with a self-issued card authenticate option.

Signature validation key properties

Validate signatures on Infocard tokens
When checked, indicates that you must sign the Information Card tokens and then indicate what type of public key to use to validate the digital signature. Clear the check box to turn off signature validation. This check box is selected by default.

Type of signature validation key

- **Public key from the KeyInfo in the signature of the Information Card token**

  Select to use the public key from the KeyInfo in the signature of the Information Card token. This is the default selection.
- **Public key from a keystore**

Select to use a public key from a keystore. If you select this option, you must select the keystore and key.

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties
Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Properties for relying party partners for identity provider federations

Federation identification

Member of federation name
The federation to which this partner has been added. You cannot modify this property.

For example, the relying party is now a partner of the identity provider federation:

```
infocard-idp
```

Partner role
Service provider (Relying party). You cannot modify this property.

Status The Partner properties page displays a partner status of Enabled or Disabled. Partners must be enabled (activated) before they can participate in a federation. You cannot modify this property because this property applies to all relying parties.

Service provider company name
This value indicates that this partner configuration is used for all partners.

For example, for an identity provider federation named infocard-idp, the default value is:

```
All Relying Parties for infocard-idp
```

Company URL
The URL of the partner company. This field is optional. For example:

```
http://www.example.com
```

Contact person
Optional contact information for the administrator. You can use the Other information field if necessary.

Infocard global partner settings

Maximum allowable clock skew between hosts (seconds)
Specify an integer value indicating the maximum amount of allowable clock skew, in seconds, between the Relying Party host and the Identity Provider host. You must specify a minimum value of zero seconds for this field. The default value is 60. This field is only available when the federation uses the Authenticate with a self-issued card authenticate option.

Select Key for Signing Assertions
Specify the key to use for signing SAML assertions.

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use

Token properties

Include the following attribute types
Specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file or by the custom mapping module will be included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. Use && to separate multiple attribute types.

Include the InclusiveNamespaces element in the canonicalization of the assertion during signature creation
Select to use the InclusiveNamespaces element in the canonicalization of the assertion during signature creation. The default is unchecked.

Include the X509 Certificate data in the KeyInfo element of the signature
Select to use the X509 Certificate data in the KeyInfo element of the signature. The default is checked.

Include the public key data in the KeyInfo element of the signature
Select to use the public key data (X509 public RSA/DSA key) in the KeyInfo element of the signature. The default is checked. KeyInfo element contains information about the key that is needed to validate the signature.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties
Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Chapter 21. OpenID planning overview

Tivoli Federated Identity Manager supports single sign-on through use of the OpenID protocol.

This overview describes the Tivoli Federated Identity Manager implementation of OpenID. The information in the overview enables an administrator to deploy and configure single sign-on federations.

The OpenID specifications refer to an *OpenID Provider or Identity Provider* as the party who asserts that a user owns a particular identity URL. A Relying Party or *Consumer* is referred to as the party who receives that information from the identity provider. In Tivoli Federated Identity Manager, the term *identity provider* is a direct match for the OpenID concept of *OpenID Provider or Identity Provider*. The OpenID *Consumer* fits well into the Tivoli Federated Identity Manager concept of service provider.

Tivoli Federated Identity Manager support for OpenID authentication allows for all the OpenID message modes:

associate
A mode for establishing a shared secret with the consumer

checkid_immediate
A mode for performing a non-blocking check to see if a user owns the claimed identifier URL.

checkid_setup
A mode for performing a check to see if a user owns the claimed identifier URL. The check can optionally include interaction with the user.

check_authentication
A mode for determining if a message signature is valid. This mode is typically used for dumb or stateless consumers.

Note: For a complete description of the OpenID specifications, see the Open ID Web site:

http://www.openid.net

OpenID 1.1 and 2.0 support

Both OpenID 1.1 and OpenID 2.0 are supported.

OpenID ID URLs

An OpenID Identity URL is a digital identity designed to be used to authenticate users and grant access to services.

Identity URL with a WebSEAL point of contact

Use the following example values to build an Identity URL with a WebSEAL point of contact:

- An identity provider federation called `openidfedip`

- A Tivoli Federated Identity Manager server where the point of contact server is WebSEAL, with the hostname `webseal.example.com`.
- A user identity (in this case a Tivoli Access Manager user) of `john`.

The OpenID Identity URL can be any URL that meets the following requirements:

- Be resolvable to your Web site. For our example, the URL must either:
  - Start with `http(s)://webseal.example.com`
  - Or, if you are using DNS wildcard entries and a site certificate for `*.example.com`, it could be a value like `http(s)://john.example.com`
- It must contain an identifier that is unique to the user. Typically this identifier is your user identity at the identity provider, however it can be a generated alias for privacy reasons.
- It must match a regular expression that you configure for your OpenID identity provider federation.
- The OpenID identity provider endpoint must be discoverable using either Yadis or HTML discovery from your identity URL as described in the OpenID specifications.

Identity URL with a WebSphere point of contact

Use the following examples values to build the Identity URL with a WebSphere point of contact:

- An identity provider federation called `openidfedip`
- A Tivoli Federated Identity Manager server where the point of contact server is WebSphere, with the hostname `poc.example.com`
- A user identity of `john`

The same requirements apply to the URL as discussed for the first example. Figure 22 shows a sample code when a WebSphere point of contact server is deployed and HTML discovery is used.

```
<html>
<head>
<link rel="openid.server"
href="https://poc.example.com/sps/openidfedip/openid/sso">
<link rel="openid2.provider"
href="https://poc.example.com/sps/openidfedip/openid/sso">
</head>
...
</html>
```

Figure 22. Example code for returning a pointer to your OpenID server from your identity URL using HTML discovery

Note: You can also use Yadis discovery for returning a pointer to your OpenID server from your identity URL.

Example identity URL

When you configure a federation for OpenID, set a regular expression for identity URLs. An easy way to ensure that you can return a link to your OpenID server endpoint from the page returned from your identity URL is to:

1. Ensure that identity URL page is an unprotected page.
2. Embed the OpenID server link in the login form for the point of contact server.

The point of contact server is typically WebSEAL or WebSphere.

This method has the limitation that there can be only one OpenID identity provider federation on the computer. Normally, this restriction causes no problems, and matches the typical deployment of OpenID.

Examples:

- For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/john
```

This simple method of configuration requires no interaction with the user to establish the identity URL. Tivoli Federated Identity Manager determines if a user owns this identity URL by:

1. Replacing the `@ID@` macro in the configured regular expression with the Tivoli Federated Identity Manager username, and
2. Verifying that the identity URL claimed by the user in the single sign-on request is an exact match.

- Another deployment example is one that where the deployment:
  - Uses a site certificate with `CN=*.example.com`
  - Uses a DNS wildcard entry which maps `*.example.com` to a site that is protected by WebSEAL.
  - Allows user to have either `http` or `https` OpenID URLs.

  For this example, the following identity URLs are valid:
  - `john.example.com`
  - `http://john.example.com`
  - `https://john.example.com`

  Note:
  - When the protocol is not specified, as in the first example, http is used.

    The regular expression that is configured for this federation would include the wildcard hostname and multi-protocol support. For example:

    ```
    http[s]?://@ID@.example.com
    ```

    The `@ID@` macro maps to a user name.
  - In some application environments, you might want to use a trailing slash in the patterns for the identity URLs:

    ```
    webseal.example.com/john/
    ```

    Some applications add a trailing slash (/) when normalizing user entry. A mismatch occurs when a trailing slash is added by the application but not specified for the identity URL. Access is not granted.

    In these environments, ensure that the configured regular expression includes the trailing slash. For example:

    ```
    http://webseal.example.com/@ID@/
    ```

Private Personal Identifier Generator

In some authentication scenarios, you might want to maintain the privacy of the user by hiding their identity from the relying party. In addition, you might also want the same user to log in to two different relying parties using different claimed identifiers. The Private Personal Identifier (PPID) Generator creates the identifier. The relying party is prevented from colliding user identities by using different claimed identifiers.

This type of authentication scenario is called *directed identity*. Directed identity requires the user to initiate login at the relying party using a shared identity provider identifier. For example `https://example.ibm.com`

Depending on the configuration, the OpenID Provider generates an identifier for the user of a specific relying-party. A Private Personal Identifier (PPID) Generator creates the identifier. The OpenID Provider generates a separate identifier for each relying party to which the same user authenticates. Creating different claimed identifiers prevents information sharing between relying-parties. This feature also effectively protects the identity of the user.

Using the Private Personal Identifier Generator feature requires the OpenID Provider to advertise its server endpoint information using an Extensible Resource Descriptor Sequence (XRDS) document. The XRDS document is required. A user can only log in to a relying party using an identity provider identifier that is discoverable through the XRDS document. The XRDS document is the only way for the relying party to differentiate between the claimed identifier of a user and an identity provider identifier.

A plug-in provides a Private Personal Identifier Generator in the identity provider implementation. The plug-in provides several standard generator implementations. An administrator can also use the plug-in to write and integrate a custom IDGenerator. This feature determines how to generate the identity of a claimed identifier for a particular user at a particular relying party.

When an identity provider identifier is used at the relying party to initiate authentication, the identity provider is responsible for generating the claimed identifier for the user. Tivoli Federated Identity Manager generates a claimed identifier using a simple configured pattern URL. The URL must contain the `@ID@` macro. The value of the `@ID@` is generated by the PPID generator.

For example, the default configuration is:

```
https://myidp.com/@ID@
```

The following IDGenerators can be used to replace the `@ID@` macro of the identity URL:

- Username ID Generator
- Hash ID Generator
- Alias service ID Generator

Username ID Generator

When the Username ID Generator is used, a username is returned as the `@ID@` portion of the expression for identity URLs.

For example, when the identity URL expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/john
```

This setting is the default behavior of Tivoli Federated Identity Manager.

Hash ID Generator

The Hash ID Generator replaces the `@ID@` value with a sha256 hash value. This hash value is a combination of the current federation ID, the username, and the relying party trust root.

The benefit of hash mode is that the username is not exposed to each site used for OpenID single sign-on. The hash value is fast to generate with no external lookups. This hiding of the account name helps to protect the user from malicious hackers who are intent on identity theft. Exposing the account name provides a staring point for phishing attacks or for locking a user from an account.

For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/
3d0f1d5e9a3a617771608b390b5c7fc1601a3839f161060cbad8e93b98f034c2
```

Alias service ID Generator

The Alias Service implementation automatically assigns a randomly generated UUID for the `@ID@` value. On first use, a UUID is generated and stored in the alias service. The lookup key for the UUID is based on the username, the current federation ID, and the relying party trust root. On subsequent uses, the same UUID is retrieved from the alias service. This method ensures that a consistent identifier is used for the user at that particular relying party.

Like the hash mode, the username is not exposed to each site used for OpenID single sign-on.

For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is: `http://webseal.example.com/c84911b2-0124-14f0-991a-a5a8f0e6f99d`

Avoiding reuse of user identities for identity URLs

OpenID identity URLs must never be reused. Once a URL has been assigned to an individual user, it must never be reassigned to another user. This specification is important because any consumer Web site to which the original user has authenticated can still have an account associated with the URL.

The requirement to ensure that OpenID identity URLs are never reused must be enforced by the deployment environment. Tivoli Federated Identity Manager cannot check for reuse. The provisioning of user names must follow a process that ensures that each URL is allocated only once.

Identity provider federations

OpenID identity provider federations share similarities with other single sign-on federations supported by Tivoli Federated Identity Manager. However, the concepts of *federation* and *partners* are applied differently. A key difference is that an OpenID identity provider does not need to know about the consuming party in advance. Shared secret negotiation is part of the protocol, and no pre-configuration of keys or partners is necessary.

In OpenID, the user is involved in the decision on whether to trust particular consuming partners. The decision is made by examining the *trust_root* URL on the consent-to-authenticate page. This means that the concept of partner service providers is unnecessary.

Note: In OpenID 2.0, the *trust_root* is called a *realm*.

The federation configuration contains some partner configuration properties, but these properties are used by the token modules for the security token service.

The OpenID federation naming follows the standard Tivoli Federated Identity Manager naming convention for a unique identifier or `protocolID`. The syntax is:

```
https://<hostname:port>/FIM/sps/<federation_name>/openid
```

For example:

```
https://www.example.com/FIM/sps/openidfedip/openid
```

Single sign-on endpoint

The single sign-on endpoint is the OpenID server URL. It supports requests from the consumer and from the browser, when redirected by the consumer. This URL requires unauthenticated access, in order that queries from anonymous consumer clients can be made for the following message modes:

- associate
- checkid_immediate
- check_authentication

When a checkid_setup request is received, and the user has not previously trusted the consumer, this URL also supplies the *consent-to-authenticate* prompt.

This endpoint returns authentication results to consuming sites.

Example endpoint:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/sso
```

Authentication endpoint

When a user has not previously logged into an identity provider, the single sign-on endpoint redirects the browser to this authentication endpoint. The user is then authenticated. This endpoint is required during checkid_setup operations when the user has not already authenticated to the identity provider, and single sign-on is initiated from a consumer.

When authentication succeeds, the end point typically redirects the user back to the single sign-on endpoint for further processing. The redirect is provided as a query-string parameter. The syntax is:

```
<protocolID>/authn?return=<url>
```

For example, as one continuous string:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/authn?return=
      https://webseald.example.com/FIM/sps/openidfedip/sso
```

Site management endpoint

When the identity provider receives a checkid_setup message, the identity provider asks the user for permission (consent) to provide the consumer authentication and attribute information for the user.

The identity provider uses a page template, and a browser cookie for that user to remember the user preferences. The identity provider must be able to retrieve the saved preferences in order to successfully answer messages in the checkid_immediate message mode, and to automate single sign-on responses for checkid_setup mode.

Tivoli Federated Identity Manager saves user preferences through use of a trusted sites manager extension point. The extension point uses a pluggable interface, which enables administrators to replace the default extension implementation with a custom implementation that, for example, supports a server-side storage model. Another purpose for this extension point is that a custom implementation might be used to auto-consent all trust decisions in a closed authentication environments.

The identity provider uses the trusted site manager to manage trusted and untrusted consumer sites. When the site manager asks the user to provide consent to authenticate, the user can specify policy for the specified consumer, as follows:

- Always Allow
- Allow Once
- Deny Once
- Always Deny

The user can later use the site manager to access and modify the saved preferences. Users can optionally remove a permanently trusted site or untrusted site from the list. When a user does this, the user is prompted with consent-to-authenticate on the next attempted single sign-on to that consumer.

The trusted site manager also remembers any optional attributes requested by a service provider (if the user has allowed the attributes to be shared).

The endpoint completes the following tasks:

1. Uses an HTML template to prompt the user with their set of permanently remembered trusted and untrusted sites.
2. Allow the user to remove sites from the permanent list

The syntax for the endpoint URL is:

```
<protocolID>/sites
```

For example:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/sites
```

Identity provider trust chains

In the OpenID model, the consumer can require that specific attributes be provided for each user identity. Tivoli Federated Identity Manager uses a trust service chain on the identity provider to obtain the attributes and place them in a simple XML token.

When a user contacts the identity provider by presenting an OpenID identity URL, the identity provider verifies the user identity. When the identity provider is operating in either checkid_immediate or checkid_setup mode, the trust service is invoked to acquire and populate the attribute data. In addition, the trust service is used to validate that any requested Provider Authentication Policy Extension (PAPE) authentication requirements have been met.

The identity provider uses a chain of trust service modules primarily to enable the retrieval of required and optional attribute values. The trust chain follows the standard module flow:

1. Validate

   The validate operation is performed on an IVCred token that is generated from the authentication credential for the user.
2. Map

   The mapping module can be any of the supported module types. When attribute data for the user can be extracted from the IVCred input token, an XSLT mapping rule is often a good option. A Tivoli Directory Integrator mapping module, or a custom Java mapping module, is useful when the attribute data must be obtained from an external source.
3. Issue

   The issue operation produces a Security Token Service Universal User (STSUU) token. The token supplies the set of required and optional attributes to the single sign-on protocol service, along with validated PAPE authentication information. This operation enables the service to generate an OpenID login response, or prompts again for authentication if needed to satisfy additional requested PAPE policies.

In order for the mapping module to populate required and optional attributes, it needs to know the list of required and optional attributes. The list of required and optional attributes are sent to the trust service in claims. The requested PAPE information is also available to the mapping rule in claims information.

The list of claims can also contain user preference data. For example, a persona index can be posted in the consent-to-authenticate form. The index is retrieved from the trusted consumers management extension point, and included in the claims.

```
<fimopenid:OpenIDClaims
    xmlns:fimopenid="urn:ibm:names:ITFIM:openid"
    xmlns:fimpape="urn:ibm:names:ITFIM:openid:PAPE"
    xmlns:fimqs="urn:ibm:names:ITFIM:queryservice"
    ClaimedId="http://specs.openid.net/auth/2.0/identifier_select"
    DiscoveredIdentifier="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
    IdentityURL="http://specs.openid.net/auth/2.0/identifier_select"
    IsOPIdentifierLogin="true"
    IsRPReturnToValidated="false"
    OPLocalId="http://specs.openid.net/auth/2.0/
identifier_select"
    OpenIDServerURL="https://www.myidp.ibm.com/
FIM/sps/openididp/openid/sso"
    PolicyURL="http://www.ibm.com"
    ReauthCount="0"
    ReturnTo="https://www.myrp.ibm.com/sps/myrp/openid/
loginreturn?nonce=uuid85d96a6f-0127-1f6e-bafb-c3b7deb3ed5d"
    TrustRoot="https://www.myrp.ibm.com/"
    Userdata=""
    Version="http://specs.openid.net/auth/2.0">
  <fimopenid:PrincipalName>shane</fimopenid:PrincipalName>
  <fimqs:RequestedAttributes>
    <fimqs:Attribute name="openid.sreg.email" optional="false" />
    <fimqs:Attribute name="openid.sreg.nickname" optional="true" />
    <fimqs:Attribute name="openid.sreg.fullname" optional="true" />
  </fimqs:RequestedAttributes>
  <fimpape:OpenIDPAPEClaims>
    <fimpape:Attribute name="openid.pape.preferred_auth_levels">
      <fimpape:Value>urn:ibm:names:ITFIM:5.1:accessmanager</fimpape:Value>
    </fimpape:Attribute>
    <fimpape:Attribute name="openid.pape.preferred_auth_policies">
      <fimpape:Value>http://schemas.xmlsoap.org/ws/2005/05/
identity/claims/privatepersonalidentifier</fimpape:Value>
      <fimpape:Value>http://www.idmanagement.gov/schema/2009/05/
icam/openid-trust-level1.pdf</fimpape:Value>
    </fimpape:Attribute>
  </fimpape:OpenIDPAPEClaims>
</fimopenid:OpenIDClaims>
```

Figure 23. Example claims during the identity provider invocation of the trust service

Figure 23 shows an example of claims being passed into the trust service. Note the optional claims in the RequestedAttributes list:

- openid.sreg.email
- openid.sreg.nickname
- openid.sreg.fullname

When attributes are required, the optional value is set to false.

In the example, notice that the property userdata is an empty string. This empty string indicates that there was no optional data defined during the consent-to-authenticate. This data (and the reading of it in the mapping module of the chain) is where persona-specific attribute retrieval can be accomplished for a user.

Handling large amounts of user attribute data

OpenID authentication works with URL redirects. When an authentication response from the identity provider must contain more than 2 kBof user registry data, Tivoli Federated Identity Manager automatically switches to POST messages. This behavior supports the OpenID 2.0 specification, which allows for auto-posting POST transactions for indirect messages.

Note: The automatic switch to POST messages is not supported in OpenID 1.1 deployments.

Relying Party Discovery

Using Relying-Party (RP) discovery, OpenID Providers can detect and verify the `return_to` addresses of realms that support OpenID.

Relying-Party discovery is performed when an OpenID Provider receives a solicited single sign-on request. Relying-Party then performs the discovery process on the URL specified in the `openid.realm` parameter of the sign-on message. Relying parties must publish their `return_to` URL in XRDS.

An administrator can configure the identity provider properties panel to enforce successful Relying-party discovery. There is also a macro in the consent.html page which allows an identity provider to indicate that the Relying-Party discovery has not been performed. For more information about consent to authenticate page see, "Template page for consent to authenticate" on page 296

This specification enables OpenID Providers to verify authentication requests and ensure that responses are redirected to valid `return_to` endpoints.

If discovery cannot verify the `return_to` URL on the Relying Party realm Tivoli Federated Identity Manager either displays an error or warning, depending on the configuration.

The IsRPReturnToValidated claim attribute tells the mapping rule if the `return_to` URL validation occurred. Tivoli Federated Identity Manager adds this attribute to the OpenIDClaims element passed to the security token service. It enables a mapping rule to detect when Relying-Party discovery fails and performs an appropriate action. The value for this claims attribute can be `true` or `false`.

Authentication modes

OpenID support two authentication modes:

- checkid_immediate
- checkid_setup

The checkid_immediate authentication mode is typically used in rich client environments where a smart widget wants to complete the following tasks:

- Determine whether a browser user owns a particular claimed OpenID URL
- Avoid having the browser interact with the user

To initiate a checkid_immediate request from the consumer to an identity provider, add this input parameter in the login form:

```
<input type=''hidden'' name=''openid.mode'' value=''checkid_immediate''>
```

The Tivoli Federated Identity Manager single sign-on protocol URL endpoint initiates the login from the consumer.

- When the response from the identity provider is a successful assertion that the user owns the identity URL, Tivoli Federated Identity Manager performs a security token service token exchange, and login to the point of contact server. This behavior is the same as for checkid_setup.
- When the response from the identity provider is a failed assertion, an HTML page template is loaded from the page factory. The replacement macro in the page is populated with the value of the `open.user_setup_url` parameter that was returned from the identity provider.

The checkid_setup authentication mode allows the identity provider to interact with the user, to request authentication or self-registration before returning a result to the consumer. When no authentication mode is specified in the login form, checkid_setup is the default mode.

Since checkid_setup is the default mode, it is not necessary to specify the mode in the login form. However, the consumer can specifically request this mode. The code to do this is:

```
<input type=''hidden'' name=''openid.mode'' value=''checkid_setup''>
```

Tivoli Federated Identity Manager support for checkid_setup is a federated single sign-on flow with redirect to the identity provider for authentication, including user interaction for approval of sign-on. The result of the authentication flow is the return of signed response attributes to the consumer. When the digital signature is validated, the attributes are be built into an Security Token Service Universal User (STSUU) token and sent to the trust service for exchange for an IVCred credential. The credential is then used for the login.

Consumer federations

The Tivoli Federated Identity Manager OpenID *consumer* plays a role like a *service provider* in other single sign-on protocols. The OpenID consumer uses a Tivoli Federated Identity Manager federation that has some similarities to, but significant differences from, the federations for other single sign-on protocols.

In particular, there is no need to directly associate identity provider partners with OpenID consumer federations. The key exchange and association with particular identity providers is controlled by the OpenID identity URL, as determined at runtime. Partners are not added and configured for an OpenID service provider federation.

The Tivoli Federated Identity Manager federation entity for the consumer contains:

- A login endpoint
- A login return endpoint
- A trust root URL (known as a *realm* in OpenID 2.0)
- Parameters indicating the type of map module in the trust chain
- Any associated configuration parameters
- User-agent policy controlling the allowed range of IP addresses, networks, and (or) hostname patterns for OpenID identity URLs and OpenID server endpoints.

The syntax for the OpenID federation protocolID is:

```
https://<hostname:port>/FIM/sps/<federation name>/openid
```

For example:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid
```

The login endpoint

The Tivoli Federated Identity Manager consumer supports a login URL. The login URL receives the POST of the initial login form and initiates a checkid_setup or checkid_immediate.

Using the previous example federation protocolID, the endpoint is:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid/login
```

Note: An example endpoint for deployments with WebSphere as point of contact server is:

```
https://poc.example.com/sps/openidfedsp/openid/login
```

The single sign-on delegate at the endpoint completes the following tasks:

1. Determines from the incoming login form the OpenID identity URL plus any extension parameters.
2. Determine the canonical form of the identity URL, in accordance with the applicable OpenID authentication specification. Yadis and HTML discovery are supported.
3. Retrieve the final identity URL, including delegates, for the user. Determine the OpenID server for the user.
4. When an association does not exist with an identity provider, establish one.
5. Build a checkid_setup or checkid_immediate request to the OpenID server and redirect the browser to the identity provider.

The login return endpoint

Tivoli Federated Identity Manager support a login return URL. The browser is redirected by the identity provider to this URL after single sign-on processing is complete. This endpoint is passed as the openid.return_to parameter during the single sign-on request.

For example, this endpoint would be:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid/loginreturn
```

The single sign-on delegate at this endpoint processes responses from checkid_setup and checkid_immediate. The delegate processes these responses, and any check_authentication requests or association handle invalidation that might occur as a result.

- When a response is returned with a successfully validated signature, the trust service uses the parameters in the response. The parameters are used to build a Security Token Service Universal User (STSUU) token. The delegate uses the trust service to exchange the STSUU token for an IVCred credential. The credential is then used for Tivoli Federated Identity Manager authentication.
- When the response is returned with an unsuccessful response, an error page is displayed.

The trust root or realm URL

The Tivoli Federated Identity Manager consumer also supplies a *trust root* or *realm* URL. This URL serves as the basis for trust displayed to the user at the identity provider.

Tivoli Federated Identity Manager reads the trust root URL from configuration properties. This property is initially generated by entries done by the administrator to combine the following values:

- Protocol

  For example, `https`.
- Hostname

  Host name for the point of contact server
- Port

  Optional. Specified only when not the standard port.
- A forward slash (/)

For example:

```
https://webseald.example.com/
```

OpenID login

The Tivoli Federated Identity Manager consumer presents a login form to request the OpenID URL from the user. The form can use either POST or GET methods to the Tivoli Federated Identity Manager consumer login endpoint. The included parameters can contain more than the URL if required.

Tivoli Federated Identity Manager supports:

- OpenID 1.1 authentication specifications
- OpenID 2.0 authentication specifications
- OpenID Simple Registration Extension 1.0
- OpenID Simple Registration Extension 1.1
- OpenID Attribute Exchange Extension
- Provider Authentication Policy Extension 1.0

Note: The method for login by the Tivoli Federated Identity Manager consumer is the same when accessing either a Tivoli Federated Identity Manager identity provider, or another identity provider.

For example, consider the following deployment scenario:

- WebSEAL as the point of contact for a host called `www.example.com`
- An OpenID consumer federation called `openidfedsp`

Figure 24 shows a sample login form for this example.

```
<html>
  <form method="post"
  action="https://www.example.com/FIM/sps/openidfedsp/openid/login">
    <img src="login-bg.gif" />&nbsp;
    <input type="text" name="openid_identifier" />&nbsp;
    <input type="submit" value="Login" />
  </form>
</html>
```

Figure 24. Simple OpenID login form

The Tivoli Federated Identity Manager service provider completes the following steps:

1. Reads the `openid_identifier` parameter
2. Performs the authentication flow specified for OpenID Authentication 2.0
3. Performs an External Authentication Interface (EAI) login to WebSEAL

After a successful checkid_immediate or checkid_setup response, the Tivoli Federated Identity Manager consumer calls the trust service to perform any required attribute or user identity manipulation.

During the login process, the consumer can request attributes from the identity provider by specifying additional parameters in the login form. The parameters must correspond to the parameter names described in the OpenID Simple Registration Extension 1.0. You can also use other supported specifications such as Simple Registration Extension 1.1, Attribute Exchange 1.0 and Private Personal Identifier Generator 1.0.

For example, Figure 25 shows a login form that accomplishes the following requirements using Simple Registration Extension:

- Requires the e-mail address from the identity provider
- Requires the date of birth from the identity provider
- Optionally requests the full name for the user
- Provides a policy URL that links to a page that describes a privacy policy

```
<html>
  <form method="post"
  action="https://www.example.com/FIM/sps/openidfedsp/openid/login">
    <input type="hidden" name="openid.sreg.required"
     value="email,dob" />
    <input type="hidden" name="openid.sreg.optional"
     value="fullname" />
    <input type="hidden" name="openid.sreg.policy_url"
value="http://www.example.com/privacy_policy.html" />
    <img src="login-bg.gif" />&nbsp;
    <input type="text" name="openid_identifier" />&nbsp;
    <input type="submit" value="Login" />
  </form>
</html>
```

Figure 25. OpenID login form with registry extension parameters

When these parameters are present in the login request, Tivoli Federated Identity Manager sends them to the identity provider. This action is done during checkid_immediate and checkid_setup requests.

The parameters do not have to be hidden, and do not have to be a comma-separated list.

The parameters can consist of multi-valued attributes. The use of multi-valued attributes enables the server to present the user with radio buttons, list boxes, or other multi-valued widgets in the HTML. Tivoli Federated Identity Manager treats each value as a comma-separated list. Multiples values consisting of one entry only (each) are allowed.

You can implement login with automatic redirection to a specified URL. When WebSEAL is the point of contact server, the rules for processing EAI authentication apply. You can include an optional `TARGET` parameter in the login form, to redirect the user after successful authentication.

Template pages

The Tivoli Federated Identity Manager consumer uses several template HTML pages when processing authentication requests and errors:

- When the consumer processes a checkid_immediate request, and the identity provider cannot determine if the OpenID URL for the user is valid, the consumer returns a template file.

  See "Template page returned for checkid_immediate" on page 306.
- The consumer uses a template file as part of supporting POST transport for large indirect messages. The Tivoli Federated Identity Manager consumer supports POST transport for large indirect messages. The support uses a template file.

  See "Template page for OpenID 2.0 indirect post" on page 304.
- When a checkid_immediate or checkid_setup request results in an error, the consumer uses a template file to return an error.

  See "Template page returned for server error" on page 307.
- When an error occurs on the consumer that halts processing, the consumer uses a template file to return the error.

  See "Template page for OpenID error" on page 303.

Consumer trust chains

During the login process, Tivoli Federated Identity Manager handles attribute and identity mapping.

When a checkid_immediate or checkid_setup response comes back from the identity provider and reports a successful assertion, Tivoli Federated Identity Manager builds all the attributes and PAPE response data returned from the identity provider into a Security Token Service Universal User (STSUU) token, and uses the trust service to exchange that token for an IVCred credential.

The trust chain consists of:

- An STSUU Token in validate mode

  The token contains the OpenID identity URL plus any extension parameters including user attributes. The STSUU token is built by the OpenID login return delegate when it has verified the signature on a login response from the identity provider.
- A mapping module

  The use of the map module allows the consumer to perform any necessary identity and attribute mapping.

  The type of mapping module to use for the consumer federation is set when the federation is configured. The standard map module types are supported:

  - XSLT or Javascript mapping rules
  - Tivoli Directory Integrator mapping module
  - Custom mapping modules.

  In many cases, the use of scripted mapping rules is sufficient, since typically there are no external attributes to retrieve.

  The Tivoli Federated Identity Manager product distribution includes sample scripted mapping rules and a sample Tivoli Directory Integrator assembly line (mapping module).
- A IVCred credential in issue mode

Account linking

An important consumer scenario for OpenID is to perform account linking. For example, when a user has authenticated directly to a Web site which is also an OpenID consumer, the Web site may enable the user to link their account with an OpenID. By performing an OpenID login while the user is already logged in to the Web site, the consuming site can associate that OpenID with the current logged in account.

To support this scenario, Tivoli Federated Identity Manager sends claims in a WS-Trust call to the security token service. The call includes the current logged in username when an authenticated session exists.

Figure 26 on page 271 shows an example of the claims format sent to the trust service. This is available to map module implementors as part of the STSUU.

```
<fimopenid:OpenIDClaims
xmlns:fimopenid="urn:ibm:names:ITFIM:openid"
xmlns:fimpape="urn:ibm:names:ITFIM:openid:PAPE"
ClaimedId="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
IdentityURL="https://www.myidp.ibm.com/FIM/op"
IsOPIdentifierLogin="true"
IsRPReturnToValidated="false"
NormalizedIdentityURL="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
OPLocalId="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
OpenIDServerURL="https://www.myidp.ibm.com/FIM/
sps/openididp/openid/sso"
ReauthCount="0"
ReturnTo="https://www.myrp.ibm.com/sps/myrp/openid/
loginreturn?nonce=uuid85e85b2a-0127-1286-99d4-d5e72a774a5f"
Signed="openid.op_endpoint,openid.return_to,
openid.response_nonce,openid.assoc_handle,
openid.claimed_id,openid.identity,
openid.sreg.dob,openid.sreg.gender,
openid.sreg.email,openid.sreg.language,
openid.sreg.timezone,openid.sreg.fullname,
openid.sreg.postcode,openid.sreg.country,
openid.sreg.nickname,openid.ns.sreg,
openid.ns.pape,openid.pape.auth_time,
openid.pape.auth_policies,openid.pape.auth_level.ns1,
openid.pape.auth_level.ns.ns1"
Target="https://www.myrp.ibm.com/fimivt/protected/ivtlanding.jsp"
Version="http://specs.openid.net/auth/2.0">
<fimpape:OpenIDPAPEClaims>
<fimpape:Attribute name="satisfied_auth_age">
<fimpape:Value>true</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="openid.pape.preferred_auth_levels">
<fimpape:Value>urn:ibm:names:ITFIM:5.1:accessmanager
</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="satisfied_auth_policies">
<fimpape:Value>true</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="openid.pape.preferred_auth_policies">
<fimpape:Value>http://www.idmanagement.gov/schema/
2009/05/icam/openid-trust-level1.pdf</fimpape:Value>
<fimpape:Value>http://schemas.xmlsoap.org/ws/2005
/05/identity/claims/privatepersonalidentifier</fimpape:Value>
</fimpape:Attribute>
</fimpape:OpenIDPAPEClaims>
</fimopenid:OpenIDClaims>
```

Figure 26. OpenID claims during a Consumer WS-Trust call

The PrincipalName attribute in the claims, when present, contains the Tivoli Federated Identity Manager username of the currently authenticated user. This enables trust chains, through the use of mapping rules, to automatically associate a particular OpenID with an existing account.

The example contains other claims which are parameters from the single sign-on response from the OpenID server. The Identity URL attribute is what the user presented in the login form as the identity URL. The NormalizedIdentityURL attribute is the canonical form of the identity URL that results from normalization that occurs as part of the discovery process.

The STSUU token sent with the request to the trust service contains attributes for each of the listed components of the `Signed` set of attributes, plus any other query string parameters.

Figure 27 shows the STSUU generated from the WS-Trust call shown in Figure 26 on page 271.

```
<?xml version="1.0" encoding="UTF-8" ?>
<stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
<stsuuser:Principal><stsuuser:Attribute name="name"><stsuuser:Value>https://www.myidp.ibm.com/FIM/
op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value></stsuuser:Attribute>
</stsuuser:Principal><stsuuser:AttributeList><stsuuser:Attribute name="openid.identity">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute></stsuuser:AttributeList><stsuuser:RequestSecurityToken />
<stsuuser:ContextAttributes><stsuuser:Attribute name="openid.op_endpoint">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/sps/openididp/openid/sso</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.email">
<stsuuser:Value>jsmith@ibm.com</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sig"><stsuuser:Value>NuKNV1ypZC16d3og6HbvjbCedPVjhRbWAWZ9Gq6g1DU=
</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.pape.auth_level.ns1"><stsuuser:Value>1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.claimed_id">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns">
<stsuuser:Value>http://specs.openid.net/auth/2.0</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.language">
<stsuuser:Value>en</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sreg.fullname"><stsuuser:Value>John Smith</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="nonce">
<stsuuser:Value>uuid85e85b2a-0127-1286-99d4-d5e72a774a5f</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.pape.auth_time">
<stsuuser:Value>2010-03-22T12:47:41Z</stsuuser:Value> </stsuuser:Attribute>
<stsuuser:Attribute name="openid.return_to"><stsuuser:Value>https://www.myrp.ibm.com/sps/myrp/
openid/loginreturn?nonce=uuid85e85b2a-0127-1286-99d4-d5e72a774a5f</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.signed"><stsuuser:Value>
op_endpoint,return_to,response_nonce,assoc_handle,claimed_id,identity,sreg.dob,
sreg.gender,sreg.email,sreg.language,sreg.timezone,sreg.fullname,
sreg.postcode,sreg.country,sreg.nickname,ns.sreg,ns.pape,pape.auth_time,
pape.auth_policies,pape.auth_level.ns1,pape.auth_level.ns.ns1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.nickname">
<stsuuser:Value>Smithy</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.identity">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/
op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns.sreg">
<stsuuser:Value>http://openid.net/extensions/sreg/1.1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.pape.auth_level.ns.ns1">
<stsuuser:Value>urn:ibm:names:ITFIM:5.1:accessmanager</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.dob">
<stsuuser:Value>1980-12-25</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sreg.postcode"><stsuuser:Value>99999</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.assoc_handle">
<stsuuser:Value>uuid85ca8353-0127-1776-9b7b-c75a4586c507</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.country">
<stsuuser:Value>AU</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.pape.auth_policies">
<stsuuser:Value>http://schemas.xmlsoap.org/ws/2005/05/
identity/claims/privatepersonalidentifier http://www.idmanagement.gov/schema/2009/05/icam/
openid-trust-level1.pdf</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.mode"><stsuuser:Value>id_res</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.timezone">
<stsuuser:Value>Australia/Brisbane</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns.pape">
<stsuuser:Value>http://specs.openid.net/extensions/pape/1.0</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.gender">
<stsuuser:Value>M</stsuuser:Value> </stsuuser:Attribute>
<stsuuser:Attribute name="openid.response_nonce">
<stsuuser:Value>2010-03-22T12:48:08Zuuid85ea7849-0127-1515-b2b5-e9223d6c6970
</stsuuser:Value></stsuuser:Attribute></stsuuser:ContextAttributes>
<stsuuser:AdditionalAttributeStatement />
</stsuuser:STSUniversalUser>
```

Figure 27. Example STSUU during trust service request at the OpenID Consumer

User agent policy

.

The Consumer uses a *user agent* (HTTP client) to connect directly to OpenID identity URLs, and to the OpenID server URLs they reference. The Provider also uses the same type of user agent policy configuration for relying-party discovery operations.

The user agent can be configured to restrict the set of locations that it tried to access. This setting is done to prevent malicious users attempting to make the user-agent connect to internal resources.

The restrictions are managed through the use of a static connection policy configuration and a customizable dynamic endpoint authorization module. The dynamic endpoint module can be used to further restrict access to endpoints at runtime.

Each Tivoli Federated Identity Manager federation has one global policy setting. The setting defines default behavior when the host is not explicitly found in the allow or deny access lists. This setting either permits or denies access to URLs as a default behavior. In addition to the global policy, administrators can create custom dynamic endpoint access plug-ins. These plug-ins can check an online list of trusted or untrusted endpoints. Administrators can add the custom plug-ins to the list of dynamic endpoint access authorization modules.

Static connection policy

With a static connection policy, an administrator can list allowed and denied hosts. Depending on the selected default behavior, the administrator also can specify a list of hosts in the allow or deny list.

When the default behavior selected is **deny**, only hosts in the **allow** lists can be accessed by the user agent. This setting is restrictive. Every OpenID identity URL and server for which you want to allow access must be covered in the allow lists. When the default behavior set to **deny**, any *deny access lists* are not useful. By default all hosts are denied unless they are explicitly included in an allow list.

When the default behavior selected is **allow**, the host is contacted, unless it is included in a *deny list*. This setting is more liberal and generally enables users to log in from any legitimate OpenID server on the Internet. However, when the default setting selected is **allow**, the deny lists must be carefully configured.

Tivoli Federated Identity Manager supports the following types of lists:

Allow lists:
- A user-configurable list of hostname regular expressions
- A user-configurable list of IP address netmasks (IPv4 and IPv6)

Deny lists:
- A user-configurable list of hostname regular expressions
- A user-configurable list of IP address netmasks (IPv4 and IPv6)
- A built-in list of default-deny hostname regular expressions
- A built-in list of default-deny IP address netmasks

Note: The allow lists take precedence over the deny lists.

The access lists for host names follow the standard Java Regular Expression syntax as defined by the Pattern class. The list uses regular expressions to match the host names.

The built-in deny lists cannot be changed by users. However, the list can be overridden to allow certain entries by adding the hostname regular expressions or netmasks to the user-configurable allowed lists.

Figure 28 shows the default-deny host names. The default values provide protection against attacks that try to access arbitrary URLs on the local system

```
.*\.localdomain
localhost
```

Figure 28. Default-deny hostname regular expressions

Figure 29 shows the default-deny IP address netmasks. These netmasks include various non-routable IPv4 and IPv6 addresses. This list can be overridden by adding the networks that you want to allow connection to the allowed list of IP netmasks.

```
0.0.0.0/8
10.0.0.0/8
127.0.0.0/8
169.254.0.0/16
172.16.0.0/12
192.168.0.0/16
255.255.255.255
::/128
::1/128
::/96
fc00::/7
fe80::/10
ff00::/8
```

Figure 29. Default-deny IP address netmasks

Dynamic endpoint access plug-in

A dynamic endpoint access plug-in is a custom module. An administrator can create the custom dynamic endpoint access plug-in to check external lists of trusted and untrusted hosts.

When an administrator selects a custom **dynamic endpoint access plug-in** in the dynamic endpoint access authorization module, the software checks specified endpoints. The specified endpoints are checked to determine if they can be trusted. You can use this setting with the allow or deny access list. However, if you set the dynamic endpoint authorization to the default access approval, the software uses only endpoints in the allow or deny lists.

Example – allowing any Internet OpenID server, deny access to 9.x.x.x intranet

- To configure this environment, the default access policy show be **allow**
- The allowed hosts list is ignored.
- The denied hosts list is ignored.

- The denied IP address netmask is 9.0.0.0/8.

  Multiple netmasks can be added if there is more than one intranet, and IPv6 equivalents must be added if the network supports both IPv4 and IPv6.

Example – only allow OpenID login from example1 and example2 companies

- To configure this environment, the default access policy show be **deny**
- The list of denied hosts and IP address netmasks are ignored.
- The allowed-hosts regular expression list is:

  ```
  .*\.example1\.com,openid\.example2\.com,openidserver\.example2\.com
  ```

Example1 OpenIDs look like `john.example1.com` and the OpenID server that the identity URLs resolves to is:

```
https://www.example1.com/openidProcessing.action
```

For example 2, OpenIDs look like `openid.example2.com/<example2_screenname>`, and this resolves to an HTML page which points to the OpenID server:

```
https://api.screenname.exmple2.com/auth/openidServer
```

The need for this URL is why both these host names appear in the list.

Example - allow any hostname containing .ibm.com string

This example shows the settings to allow a user to access any hostname containing the .ibm.com string.

- To configure this environment, select a custom plug-in
- The allowed hosts list is checked.
- The denied hosts list is checked.
- The custom dynamic endpoint plug-in is:

```
package com.tivoli.am.fim.demo.ibmaccessapproval;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.tivoli.am.fim.useragent.AccessApproval;

public class IBMAccessApproval implements AccessApproval {

 final static String CLASS = IBMAccessApproval.class.getName();

 final static Logger _log = Logger.getLogger(CLASS);

 public IBMAccessApproval() {
 }

 public boolean canAccess(Map ctx) {
  String methodName = "canAccess";
  _log.entering(CLASS, methodName, new Object[] { ctx });
  boolean result = false;
  boolean finestLoggable = _log.isLoggable(Level.FINEST);
  try {
   String endpoint = (String) ctx.get(AccessApproval.CTX_ENDPOINT);
   String fedname = (String) ctx
     .get(AccessApproval.CTX_FEDERATION_NAME);
   String fedid = (String) ctx
```

```
   .get(AccessApproval.CTX_FEDERATION_ID);

   if (finestLoggable) {
    _log.logp(Level.FINEST, CLASS, methodName, "Fedname: "
      + fedname + " Fedid: " + fedid + " Endpoint: " + endpoint);
   }

   try {
    URL u = new URL(endpoint);
    String hostname = u.getHost();
    if (hostname != null && hostname.indexOf(".ibm.com") > 0) {
     result = true;
    }
   } catch (MalformedURLException e) {
    e.printStackTrace();
   }

  } finally {
   _log.exiting(CLASS, methodName, "" + result);
  }
  return result;
 }
}
```

OpenID Extensions

OpenID Simple Registration Extension

During the login process, the consumer can request attributes from identity providers by specifying additional parameters in the login form. The parameters must correspond to the parameter names described in the OpenID Simple Registration Extension 1.0 or Attribute Exchange Extension 1.0, whichever is applicable. Simple Registration Extension (SREG) is an extension to the OpenID Authentication protocol and supports a simple list of common user registration information. For more information, see OpenID documentation at: http://openid.net/specs/openid-simple-registration-extension-1_0.html

```
<form name="openidLoginForm" method="post"
action="https://sp.example.com/FIM/sps/openidsp/openid/login">
<input name="openid.mode" type="hidden"
value="checkid_setup">
<input name="openid.sreg.required" type="hidden"
value="email">
<input name="openid.sreg.optional" type="hidden"
value="fullname,dob">
<input name="openid.sreg.policy_url" type="hidden"
value="https://sp.example.com/privacy_policy.html">
<input name="TARGET" type="hidden"
value="https://sp.example.com/myapp">
<input name="openid_identifier" type="text">
<input value="OpenID Login" type="submit">
</form>
```

Figure 30. Sample Simple Registration Extension

OpenID Attribute Exchange Extension

Identity providers can use OpenID extensions to obtain and communicate user attributes to consumers.

The attribute exchange extension provides identity providers the ability to communicate user attributes to consumers.

The Attribute Exchange Extension (AX) protocol can be extended to accommodate varying types of attributes and multi-valued attributes. The attributes are identified by a unique URI and typically correspond to personal identity information. For more information see OpenID documentation at: http://openid.net/specs/openid-attribute-exchange-1_0.html

Attribute Exchange Extension provides strict compatibility with OpenID 2.0. You can use either or both extensions simultaneously. Use Attribute Exchange Extension unless you need to be compatible with older OpenID 1.1 implementations that only support SREG.

As an administrator, you can add a set of parameters to the OpenID login form posted to the login endpoint.

The example shows a login form with the following requirements:

- Requires the e-mail address from the identity provider
- Optionally requests for the full name, date of birth, friends and groups.

```
<form name="openidLoginForm" method="post"
action="https://sp.example.com/FIM/sps/openidsp/openid/login">
<input name="openid.mode" type="hidden" value="checkid_setup">
<input name="openid.ax.required" type="hidden" value="axemail">
<input name="openid.ax.if_available" type="hidden"
value="axfullname,axdob,axfriends,axgroups">
<input name="openid.ax.type.axemail" type="hidden"
value="http://axschema.org/contact/email">
<input name="openid.ax.type.axfullname" type="hidden"
value="http://axschema.org/namePerson">
<input name="openid.ax.type.axdob" type="hidden"
value="http://axschema.org/birthDate">
<input name="openid.ax.type.axfriends" type="hidden"
value="http://example.com/myschema/friends">
<input name="openid.ax.count.axfriends" type="hidden"
value="5">
<input name="openid.ax.type.axgroups" type="hidden"
value="http://example.com/myschema/groups">
<input name="openid.ax.count.axgroups" type="hidden"
value="unlimited">
<input name="TARGET" type="hidden"
value="https://sp.example.com/myapp">
<input name="openid_identifier" type="text">
<input value="OpenID Login" type="submit">
</form>
```

Figure 31. Sample Attribute Exchange Extension

Note: If no explicit count is requested for an attribute exchange parameter, the default max count value is 1.

Tivoli Federated Identity Manager sends parameters to the identity provider during `checkid_immediate` and `checkid_setup` requests. The fetch messages sent with the request retrieves the user's personal identity attributes. For additional information about fetch messages see the OpenID documentation: http://openid.net/specs/openid-attribute-exchange-1_0.html#fetch

Attribute Exchange Extension fetch requests parameters

The Attribute Exchange Extension supports an information model that combines a subject identifier, an attribute type identifier, a count, and a value. Including additional parameters attaches the Attribute Exchange Extension fetch request on a standard authentication request. To enable the consumer to retrieve information from the identity provider, specify the following form field parameters in the login form.

openid.ax.required
: Fetches required attributes from the identity provider. The value is a list of aliases, which are labels that represent individual attributes at the identity provider. Bind each alias to a URI that identifies the attribute in a separate `openid.ax.type.`*`alias`* parameter. *(Optional)*

openid.ax.if_available
: Fetches an attribute that is available from the identity provider. The value has the same requirements as `openid.ax.required`. *(Optional)*

Note: You must specify either `openid.ax.required` or `openid.ax.if_available` in the request. Each requested attribute alias must have an associated `openid.ax.type.`*`alias`* parameter.

openid.ax.type.*alias*
: Binds the alias to a URI that defines the meaning of the attribute. You must specify a parameter for each alias specified in either `openid.ax.required` or `openid.ax.if_available`. *(Optional)*

  Many typical attributes already have defined type URIs at http://www.axschema.org/types/

openid.ax.sendalways
: Includes OpenID Attribute Exchange Extension information in authentication requests to the identity provider. The consumer runtime sends Attribute Exchange Extension request information if the identity provider advertises Attribute Exchange Extension support with XRDS. The default value is false. *(Optional)*

Attribute Exchange Extension fetch response parameters

After granting access to an identity provider, a fetch response message supplies the information in the fetch request parameters. The following optional fetch response parameters specify the retrieved personal attributes from the identity provider.

openid.ax.type.*alias*
: Specifies the URI type for the fetched attribute identified by alias. *(Optional)*

openid.ax.count.*alias*
: Returns the number of values specified for the attribute that corresponds to alias. If you do not specify a specific value, it returns only one value.

openid.ax.value.*alias*
: Assigns a value specified for the attribute that corresponds to alias. *(Optional)*

openid.ax.value.*alias***.number**
: Assigns a value specified for the attribute that corresponds to alias. This parameter is required if `openid.ax.count.`*`alias`* is sent and at least one

value is configured for the associated attribute. There should be a separate parameter for each value for the alias, with incrementing numbers.

OpenID Provider Authentication Policy Extension

When a user initiates authentication from a relying party with an OpenID identifier, the relying party requests the identity provider to authenticate the user.

The OpenID Provider Authentication Policy Extension (PAPE) is a mechanism which allows a relying party to:

- request identity providers to use specific authentication policies when authenticating a user.
- require an identity provider to inform the relying party of the authentication policies used during authentication.
- require an identity provider to communicate the levels of authentication used as defined in sets of requested custom assurance levels.

Use the administration console to configure the OpenID Provider Authentication Policy Extension. Depending on your role in the federation, certain parameters are available in the configuration properties panel.

Note: PAPE settings can only be configured AFTER creating a federation. Use the Federation Properties panel to specify the configuration settings.

Relying party PAPE implementation

Use the relying party configuration properties panel to enable PAPE. Once enabled, the specified PAPE attributes are sent to the identity provider in the authentication request. When a relying party sends the authentication request with the specified PAPE attributes, the identity provider sends a response. The response indicates which requirements have been met and which have not. Based on the response, the relying party can determine whether to authenticate a user.

Specify the following parameters in the relying party configuration properties panel:

Enforcement Mode

- Strict

  Specifies that a user is not authenticated if the PAPE requirements are not met.
- Lenient

  Specifies that a user is authenticated even if the PAPE requirements are not met. The mapping rule used in the federation accesses the response information. The response indicates which requirements have been met and which have not been met. This setting allows the author of the mapping rule to decide whether to log in the user based on that information. The mapping rule provides a more restricted authorization.

Authentication policies

Specifies a set of authentication policy URIs. The URIs represent authentication policies to be satisfied by the identity provider when authenticating a user. If multiple policies are requested, the identity provider must satisfy as many of them as it can. The identity provider then indicates which authentication policies were satisfied in the response.

Maximum authentication age

Specifies the length of time in which the user must have been authenticated. If this time has expired, the identity provider must re-authenticate the user.

Preferred assurance level

Specifies an ordered list of preferred assurance level namespace URIs. The assurance level namespace values determine the level of trust placed in the authentication of the user. Relying parties request information about these assurance level namespaces from the identity provider.

Identity provider PAPE implementation

The identity provider configuration properties panel specifies the conditions for when a user must authenticate.

Note: If you plan to use WebSEAL cookie management with OpenID PAPE implementation, ensure that the list of managed cookies does not include the WebSphere session cookie. See "Configuring WebSEAL to manage cookies" on page 391

Specify the following parameters in the identity provider configuration properties panel:

Force authentication on any requested PAPE maximum authentication age

This parameter specifies that a user must always authenticate. If selected, the Maximum authentication age allowable clock skew field is disabled.

Maximum authentication age allowable clock skew

When a maximum authentication age is requested by a service provider during single sign-on, the identity provider mapping rule must return the last authentication time of the user. This parameter is used to account for clock skew between:

- the last authentication time returned by the identity provider mapping rule
- the clock of the identity provider

Typically the skew time is a small number, but can account for differences between the point of contact machine and the runtime machine.

Identity provider configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of OpenID federations. The wizard prompts you to supply properties for your deployment. This worksheet describes the properties.

Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name

The name can be any character string. For example, `openid-idp`. This field is required.

Federation role

Your role is *identity provider*.

Company name

The name of the company that is creating this federation. The value can be any string. The space character is allowed. This field is required.

Federation protocol

OpenID.

Point of contact server

The URL address of the server that acts as initial point of contact for incoming requests. The address consists of a protocol specification, the server hostname, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified.

Example value:

```
https://webseald.example.com/FIM
```

Note: For OpenID support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Association expiration (seconds)

Specifies the lifetime of the association handle. This identity provider controls this value. Enter a positive number. The default value is 3600 seconds.

Response nonce expiration time (seconds)

Indicates how many seconds a relying party that is operating without an established association has, before they must perform the `check_authentication` request. If set to a positive number, this feature prevents replay of `check_authentication`. This restriction applies to customers with Relying Parties that cannot create or store associations. The default value is 30 seconds.

ID Generator

Specifies which ID generator creates a value that replaces the `@ID@` of an identity URL. Different ID generators create different values for `@ID@`.

OpenID Identity URL pattern

Represents the regular expression on which identity URLs are matched for the federation. Tivoli Federated Identity Manager replaces the `@ID@` part. The default value is the URL for the single sign-on protocol hostname provided by the installation wizard.

For example, if you specified the following point of contact server in the wizard:

```
https://webseald.example.com/FIM
```

the default Identity URL pattern is:

```
https://webseald.example.com/@ID@
```

User setup URL

Specifies the URL that is sent in response to a `checkid_immediate` request from a consumer. The URL is used when the identity provider is unable to determine if a user owns a particular identity URL.

The default URL is the point of contact server URL you specified on the Point of Contact Server panel.
For example, when you have previously specified, in the wizard, a point of contact server:

```
https://webseald.example.com/FIM
```

the default user setup URL is:

```
https://webseald.example.com/
```

Trusted Sites Manager
Selects the implementation class for a trusted sites manager. The implementation persists data concerning consent-to-authenticate decisions made by a user during OpenID authentications.

Support OP Identifier
Specifies if `identifier_select` is supported when a consumer initiates single sign-on. Use this option if an identity provider uses XRDS. Not selecting this option disables all other options for `identifier_select`.

OP Generated Claimed Identifier Pattern
Specifies a valid URL that must contain the `@ID@` string. It enables a relying party to initiate single sign-on with a claimed identifier set to `identifier_select`.

The default URL is derived from the point of contact server URL specified during federation configuration.

For example, if the point of contact URL was specified as:

```
https://webseal.example.com/FIM
```

the default OP Generated Claimed Identifier Pattern is:

```
https://webseal.example.com/@ID@
```

Relying-Party Discovery Options
Provides two options:

Perform RP Discovery
Specifies whether to attempt relying party discovery. Not selecting this option disables all other options for relying party discovery.

Require successful RP Discovery
Specifies if Tivoli Federated Identity Manager halts with an error when it cannot complete relying party discovery for the identity provider. This option applies only if you enable Perform RP Discovery.

RP Discovery cache expiration time
Determines how many seconds to cache information discovered about relying parties in seconds. If you enter less than zero, information is never cached.

Permitted OpenID Server Protocols
Specifies the allowed protocols for the OpenID servers to which the user agent permits connection. You can choose either or both values. For best practice, the parameter is typically is set to HTTPS only.

Choose one or both of the following:

- HTTPS
- HTTP

HTTP Connection Timeout
Specifies how many seconds before a timeout occurs during communications with the HTTP client. Enter a positive number. If you enter zero (0), the software uses the Java defaults for URLConnection objects. The default value is 30 seconds.

Keystore

Specifies the keystore used to validate the certificates of SSL endpoints during communications for relying-party discovery. This keystore must contain the certified authority signer certificates of all relying-parties for which relying-party discovery is to be performed.

User Agent Connection Policy

Specifies policy for connections by the user agent. You must select one of the following options.

- Allow access to OpenID hosts by default

  The host is contacted, unless it is included in the deny list. This setting is more liberal and generally enables users to log in from any legitimate OpenID server on the Internet.
- Deny access to OpenID hosts by default

  Only hosts in the allow lists can be accessed by the user agent. This setting is restrictive, and every OpenID identity URL and server for which you want to allow access must be covered in the allow lists.

To review the policy choices, see "User agent policy" on page 272.

Allowed Hostname Regular Expressions

Specifies a list of regular expressions that identify host names to which the user agent can request access. Enter one string per line.

For example:

```
.*\.ibm\.com
```

The value is optional.

Allowed IP Address / Netmasks

Specifies IP addresses or netmasks to which the user agent can request access. User regular expressions, and enter one string per line. Enter one string per line.

For example:

```
10.1.1.0/24
192.168.0.10
```

This value is optional.

Dynamic Endpoint Access Authorization Module

Specifies a list of custom dynamic endpoint access plug-ins. The plug-ins can check external lists of trusted and untrusted hosts. This setting is used in addition to configuration in the User Agent Connection Policy. If set to the default access approval, configuration settings specified under User Agent Connection Policy are used.

Denied Hostname Regular Expressions

Specifies the host names to which the user agent cannot request access. User regular expressions, and enter one string per line.

- When the User Agent Connection Policy is set to Deny access to OpenID hosts by default, this property is not used.
- When the User Agent Connection Policy is set to Allow access to OpenID hosts by default, use of this property is optional.

For example:

```
.*\.example\.com
.*\.example2\.com
```

Denied IP Address / Netmasks

Specifies a list of regular expressions that identify IP addresses or netmasks to which the user agent cannot request access. Enter one string per line.

- When the User Agent Connection Policy is set to Deny access to OpenID hosts by default, this property is not used.
- When the User Agent Connection Policy is set to Allow access to OpenID hosts by default, use of this property is optional.

For example:

```
11.12.13.0/24
192.168.0.10
```

Identity mapping options

Select one of the following options:

- **Use XSLT or Javascript mapping rules for identity mapping**

  Select this option when you create an XSLT or a Javascript mapping rule that supplies identity mapping rules.

  Tivoli Federated Identity Manager provides a sample identity mapping rules file for OpenID identity provider federations:

  /*installation_directory*/examples/ip_openid.xsl
- **Use Tivoli Directory Integrator for mapping**

  Select this option when you have a Tivoli Directory Integrator assembly line for the identity mapping required for your OpenID federation.
- **Use custom mapping module instance**

  Select this option when you have a custom trust service module for the identity mapping required for your OpenID federation.

Table 100. Worksheet for federation identification properties

Property to specify	Your value
Federation name	
Role	identity provider
Company name	
Federation Protocol	OpenID
Point of Contact server	
Association expiration time (seconds)	Default: 3600 seconds
Response nonce expiration (seconds)	Default: 30 seconds
ID Generator (generates value of @ID@)	
OpenID Identity URL Pattern	
User Setup URL	
Support OP Identifier	OP Generated Claimed Identifier Pattern
OP Generated Claimed Identifier Pattern	

Table 100. Worksheet for federation identification properties (continued)

Property to specify	**Your value**
Perform RP Discovery	
Require successful RP Discovery	
RP Discovery cache expiration time	
Permitted OpenID Server Protocols	
HTTP Connection Timeout	
Keystore	
User Agent Connection Policy	
Allowed Hostname Regular Expressions	
Allowed IP Addresses / Netmasks	
Dynamic Endpoint Access Authorization module	
Denied Hostname Regular Expressions	
Denied IP Addresses / Netmasks	
Identity mapping options	Select one: • Use XSLT or JavaScript for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSLT or JavaScript for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Consumer configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of OpenID federations. The wizard prompts you to supply properties for your deployment. This worksheet describes the properties.

Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example, `openid-consumer`.

Federation role

Your role is *service provider*. You must select *service provider* when configuring the consumer role.

Company name

A string value for the Company name. You can optionally provide additional contact information.

Federation protocol

OpenID.

Point of contact server

The URL address of the server that acts as initial point of contact for incoming requests. The address consists of a protocol specification, the server hostname, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified. Example value:

```
https://webseald.example.com/FIM
```

Note: For OpenID support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Advertised trust root

This value is a URL that is the *root* of the trust URLs for the federation. It defaults to the base URL for the federation, where the base URL is the path to the single sign-on protocol service hostname. When the port is not the default number, the port value is also included. The value must end in a forward slash (/).

This value must be a URL parent of the login return delegate endpoint. It is used as the openid.trust_root parameter in the single sign-on request that is sent to the identity provider.

For example, when you have previously specified, in the wizard, a point of contact server:

```
https://webseald.example2.com/FIM
```

the default authentication trust root is:

```
https://webseald.example2.com/
```

Enable Yadis Protocol

Specifies whether to perform the Yadis discovery. For best practices, choose enabled.

Enable XRI Identifiers

Specifies whether to resolve URL or XRI-based claimed identifiers. If you do not select an option, the software uses only URL-based claimed identifiers.

XRI Proxies

Specifies a list of URLs for resolving XRI identifiers. The URL must contain the `@XRI@` macro.

Discovered information expiration

Specifies how long the cache stores the discovered information. If you do not enter a positive number, the cache is disabled and discovery is performed at every login.

Response nonce skew time

Specifies a value in seconds used for validating the response nonce from OpenID 2.0 identity providers. Validation is only performed if this skew is a positive number. Validation is performed by taking the time of the response nonce and the configured response nonce skew. If the number of seconds is outside this range, the authentication response is rejected. If the number of seconds is within this range, a response nonce cache is checked. The check ensures that the authentication response is not a replay. When validation is successful, the response nonce is added to the response nonce cache for as long as it would be within the skew period. The response nonce is added to the nonce cache to ensure that future authentication responses are not replays.

Permitted OpenID Server Protocols

This value represents the set of allowed protocols for the OpenID servers to which the user agent permits connection. For best practice, the parameter is typically is set to HTTPS only.

Choose one or both of the following:

- HTTPS
- HTTP

HTTPS is the default. You must select at least one protocol.

HTTP Connection Timeout

This value specified the communications timeout for the HTTP client. The value must be a valid positive integer. The maximum value is the maximum integer value. A value of zero (0) means to use the Java defaults for URLConnection objects. The default value is 30 seconds.

Keystore

This value is the name of a keystore that has previously been configured in the Tivoli Federated Identity Manager key service. The keystore must hold certificate authority signer certificates only.

The HTTP client for the consumer uses this keystore when communicating with SSL-enabled identity providers. The keystore is used to determine if the host that is to be connected to can be trusted. This check occurs when processing `associate` and `check_authentication` messages.

Default:

```
DefaultTrustedKeyStore
```

User Agent Connection Policy

This value specifies policy for connections by the user agent. You must select one of the following options.

- Allow access to OpenID hosts by default
- Deny access to OpenID hosts by default

To review the policy choices, see “User agent policy” on page 272.

Allowed Hostname Regular Expressions

A list of regular expressions that specify host names to which the user agent can request access. Enter one string per line. For example:

```
.*\.ibm\.com
```

This value is optional.

Allowed IP Addresses / Netmasks

A list of regular expressions that specify IP addresses or netmasks to which the user agent can request access. Enter one string per line. For example:

```
10.1.1.0/24
192.168.0.10
```

This value is optional.

Denied Hostname Regular Expressions

A list of regular expressions that specify host names to which the user agent cannot request access. Enter one string per line. For example:

```
.*\.example\.com
.*\.example2\.com
```

- When the User Agent Connection Policy is set to **Deny access to OpenID hosts by default**, this property is not used.
- When the User Agent Connection Policy is set to **Allow access to OpenID hosts by default**, use of this property is optional.

Denied IP Addresses / Netmasks

A list of regular expressions that specify IP addresses or netmasks to which the user agent cannot request access. Enter one string per line. For example:

```
11.12.13.0/24
192.168.0.10
```

- When the User Agent Connection Policy is set to **Deny access to OpenID hosts by default**, this property is not used.
- When the User Agent Connection Policy is set to **Allow access to OpenID hosts by default**, use of this property is optional.

Dynamic Endpoint Access Authorization Module

Specifies a list of custom dynamic endpoint access plug-ins. The plug-ins can check external lists of trusted and untrusted hosts. This setting is used in addition to configuration in the User Agent Connection Policy. If set to the default access approval, configuration settings specified under User Agent Connection Policy are used.

Identity mapping options

You are asked to select one of the following options:

- Use XSLT or Javascript mapping rules for identity mapping

  Select this option when you have created an XSLTfile or a Javascript mapping rule that supplies identity mapping rules.

  Tivoli Federated Identity Manager provides a sample identity mapping rules file for OpenID consumer federations:

  `/`*`installation_directory`*`/examples/sp_openid.xsl`
- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your OpenID federation.
- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your OpenID federation.

Table 101. Configuration properties for OpenID consumer

Property	Your value
Federation name	
Role	service provider
Company name	
Federation Protocol	OpenID

Table 101. Configuration properties for OpenID consumer (continued)

Property	**Your value**
Point of Contact server URL	
Advertised trust root	
Enable Yadis Protocol	
Enable XRI Identifiers	
XRI Proxies	
Discovered information expiration	
Response nonce skew time	
Permitted OpenID Server Protocols	`HTTPS` or `HTTP` or both
HTTP Connection Timeout (seconds)	Default: 30 seconds
Keystore	
User Agent Connection Policy	
Allowed Hostname Regular Expressions	
Allowed IP Addresses / Netmasks	
Denied Hostname Regular Expressions	
Denied IP Addresses / Netmasks	
Dynamic Endpoint Access Authorization module	
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Chapter 22. Configuring OpenID

Verifying OpenID dependencies

Before you begin

Before you use the federation creation wizard, ensure that the OpenID dependences have been met. Complete the required planning activities by reviewing the overview material in the planning section

Procedure

1. Determine your strategy for identity mapping
   - If using a mapping rules file, ensure that the XSLT or Javascript mapping rule has been written to match the requirements of your deployment.
   - If using a Tivoli Directory Integrator assembly line, ensure that the assembly line has been constructed
   - When using a custom mapping module, ensure that the module has been written and tested
2. Ensure that you have established the user agent policy for both the consumer and identity provider.
3. Complete the worksheet for the federation. Complete one of the following:
   - “Identity provider configuration worksheet” on page 280
   - “Consumer configuration worksheet” on page 285

Configuring an OpenID federation

Use the federation wizard to create and configure an OpenID federation.

Before you begin

Ensure that you have prepared configuration information before using the wizard to create the federation.

About this task

To use the federation wizard to create and configure an OpenID federation, complete the following steps:

Procedure

1. Log in to the Integrated Solutions Console and click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets are displayed. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The wizard presents a series of configuration panels.
3. Use your completed worksheet to provide values at each panel. Supply the necessary values, and then click **Next** to proceed to the next panel. If you need to go back to adjust a configuration setting, click **Back**. You can view the online help for information about specific fields.

a. The first series of panels requests settings for the federation name, role, protocol, and point of contact server.
b. Next, the OpenID configuration panel requests the values needed for an OpenID identity provider or consumer.
c. The last series of panels requests settings for the identity mapping configuration.

When you finish entering configuration settings, the Summary panel is displayed.

4. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.

What to do next

Configuring a WebSEAL point of contact server for an Open ID federation

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the OpenID federation.

Before you begin

The Create Federation Complete portlet provides a button that you can use to obtain a configuration utility.

About this task

You must obtain the utility and run it. Complete the following steps:

Procedure

1. Click **Download Tivoli Access Manager Configuration Tool**
2. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
3. Return to the management console, and Click **Done** to return to the Federations panel.
4. Run the configuration tool from a command line. The syntax is:

```
java -jar /download_dir/tfimcfg.jar -cfgfile webseald-instance_name.conf
 -action tamconfig
```

You will need to know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command (as one continuous line) is:

```
java -jar /tmp/tfimcfg.jar
   -cfgfile /<fully_qualified_path>/webseald-default
   -action tamconfig
```

For more information, see:

- Appendix A, “tfimcfg reference,” on page 491

What to do next

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log in to the administration console.
2. Click Tivoli Federated Identity Manager > Manage Configuration > Point of Contact
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Configuring login pages

As part of configuring a point of contact server, you should configure the information on login pages.

- Consumers must provide a login form for presentation to the end user.

  Administrators who use WebSEAL as a point of contact server can choose to modify the default WebSEAL login.html page
- Identity providers need to provide discovery information using Yadis or HTML discovery. The discovery information is provided at the OpenID identity URL of the user, or the identity provider identifier URL or both.

Chapter 23. OpenID reference

Supported algorithms and transports

Tivoli Federated Identity Manager supports the OpenID specifications for the session type of the shared-secret session (association):

- OpenID 1.1
  - clear text
  - DH-SHA1

  For security reasons the Tivoli Federated Identity Manager service provider (consumer) support of OpenID 1.1 will only request session types of DH-SHA1.
- OpenID 2.0
  - DH-SHA256
  - DH-SHA1
  - no-encryption

  The Tivoli Federated Identity Manager consumer tries DH-SHA256 by default.

  When an identity provider returns an error indicating that a requested session type is unsupported, the identity provider may state which session types are supported. In this case, the Tivoli Federated Identity Manager consumer tries the suggested session type.

  Note: The Tivoli Federated Identity Manager consumer attempts to use no-encryption only when the OpenID server is an SSL endpoint.

The identity provider endpoints that are used by consumers to access OpenID should be configured to use SSL.

In most deployments, non-protected endpoints (for example, HTTP instead of HTTPS) are used for resolution of the identity URL for a user. The following URLs, which are returned as HTML header links, should use SSL:

- openid.server
- openid2.provider

The consumer endpoints should be HTTPS (SSL).

Template page for advertising an OpenID server

The OpenID authentication specifications states that when an identity provider single sign-on URL should return a notification whenever it receives with an HTTP GET request that has no parameters (as specified by the OpenID 1.1 specification). The page to be returned is required to have the following text:

```
This is an OpenID server endpoint. For more information, see http://openid.net/
```

Tivoli Federated Identity Manager provides the file openid_server.html. The file does not have any replaceable macros.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>OpenID Server</title>
  </head>
  <body>
    This is an OpenID server endpoint. For more information,
see <a href="http://openid.net/">http://openid.net/</a>
  </body>
</html>
```

Figure 32. Template file openid_server.html

This template is used on the identity provider only.

Template page for consent to authenticate

This page is used at the identity provider to determine and store user consent information about whether or not to permit authentication to a particular consumer, and to indicate which optional attributes should be shared with that consumer.

During an OpenID checkid_setup operation, the user is redirected to the Identity Provider to validate they are logged in. At this time, the identity provider asks the user for permission to provide authentication and attribute information to the consuming site. The Tivoli Federated Identity Manager identity provider provides an HTML template page called consent.html.

Tivoli Federated Identity Manager retains knowledge of decisions about whether a user trusts a particular consuming site, in the form of the `trust_root` or `realm`. This saved knowledge enables Tivoli Federated Identity Manager to not have to prompt the user every time a user logs in to the same consumer.

The consent page displays the list of attributes that the single sign-on request (from the consumer) has indicated as *required* or *optional*. Since these lists can be of indeterminate length, the template supports multiple copies of stanzas, repeated once for each attribute in either list. The support for repeated stanzas is provided through the simple registration extension specification.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

This template file provides several replacement macros:

@OPENID_TRUSTURL@

This macro is replaced with the `openid.trust_root` parameter in the checkid_setup request.

@OPENID_POLICYURL@

This macro is replaced with the `openid.sreg.policy_url` parameter in the checkid_setup request when the URL exists. When the URL does not exists, the values is an empty string.

@OPENID_IDENTITYURL@

This macro is replaced with the `openid.identity` parameter in the checkid_setup request.

@OPENID_SSOURL@
: This macro is replaced with the endpoint of the OpenID server delegate (endpoint) on the identity provider. This value is used for the FORM action parameter to post the results of the consent form back to the OpenID server.

@OPENID_RETURN_TO_VALIDATED@
: This macro is replaced with true or false to notify the user if return_to URL validation has been performed as part of Relying-Party discovery.

@REQUIRED_ATTRIBUTE@
: This is a multi-valued macro that belongs inside a `[RPT requiredAttrs]` repeatable replacement list. The values display the list of required attributes from the service provider, as specified for the simple registration extension. This macro is replaced for each value contained within the `openid.sreg.required` parameter in the request with the string `openid.sreg.` prepended.

@OPTIONAL_ATTRIBUTE@
: This is a multi-valued macro that belongs inside a `[RPT optionalAttrs]` repeatable replacement list. The values display the list of optional attributes from the service provider, as specified for the simple registration extension. This macro is replaced for each value contained within the `openid.sreg.optional` parameter in the request with the string `openid.sreg.` prepended.

Optional attributes require special consideration. The identity provider allows users to specify individually which of the optional attributes they will permit to be sent to a specified consumer. The user preferences are denoted by the true or false parameters for each optional attribute, as specified in the form contained in the HTML page for consent-to-authenticate. To enable this feature, the parameter name *must* begin with the prefix `optattr_` and end with the full name of the optional attribute.

For example:

```
optattr_openid.sreg.email=true&optattr_openid.sreg.nickname=false
```

The following figure shows an example of the handling of optional attributes.

```
 The following optional attributes have been requested. Please select
 which attributes you are prepared to send, or select
 "All Optional Attributes":<br /><br />
          <input id="chk_all_optional_attributes" type="checkbox"
           checked="checked" name="all_optional_attributes"
           onClick="allOptionalAttributes()" />
          <label for="chk_all_optional_attributes">All Optional Attributes
          </label><br /><br />
      [RPT optionalAttrs]
              <input id="chk_@OPTIONAL_ATTRIBUTE@" type="checkbox"
name="optattr_@OPTIONAL_ATTRIBUTE@" onClick="oneOptionalAttribute()" />
              <label for="chk_@OPTIONAL_ATTRIBUTE">@OPTIONAL_ATTRIBUTE@
              </label><br />
      [ERPT optionalAttrs]
```

Figure 33. Handling consent of individual optional attributes

Note that the checkbox input parameter in the form builds the name using the `optattr_` prefix, and the name of the optional attribute. For each optional attribute in the request from the service provider, the code processing this form at the identity provider look for a parameter like optattr_<attributename>, and treat the value as true or false. A value of true indicates consent of the optional attribute. When a parameter does not exist in the posted form, consent is false.

One possible deployment scenario might be the development of a *persona portal* for individual end-users. This would enable a particular use to have different personas created for them, each with different attribute sets managed in an external data store. This function enables the end user to associate a particular persona with a particular OpenID consumer. This association then allows for the selection of those particular persona attributes when the user logs into the specified consumer.

For example, an identity provider can allow a user to dynamically create, name, and populate sets of attributes for each persona.

This scenario is supported by the use of an optional FORM parameter called userdata. The userdata can be a list (selectable through a menu) that allows the user to select the persona from which the attributes should be populated.

When userdata is included in the input form, its URL-encoded string value is included in the claims sent to the security token service during identity mapping.

The following code sample shows the example HTML template file consent.html.

This template is used on the identity provider only.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>OpenID Consent-to-Authenticate</title>
<script type="text/javascript">

// when "All optional attributes" is selected,
uncheck any checked individual optional attributes
function allOptionalAttributes() {
    var theForm = document.forms[0];
    for (i = 0; i < theForm.elements.length; i++) {
        if (theForm.elements[i].type == "checkbox") {
            var cbName = theForm.elements[i].name;
            if (cbName.indexOf("optattr_") == 0) {
                theForm.elements[i].checked = false;
            }
        }
    }
}

// when an individual optional attribute is selected, be sure to
uncheck "All optional attributes"
function oneOptionalAttribute() {
    document.forms[0].all_optional_attributes.checked = false;
}

// utility function to show a section
function showDiv(f) {
  if (f.style) {
    f.style.display='block';
  }
}

</script>
  </head>
  <body>
    This consuming site has asked for an OpenID login from you:
 <b>@OPENID_TRUSTURL@</b>
    <p />
    The consuming site's policy can be found at: <b>@OPENID_POLICYURL@</b>
    <p />


<script type="text/javascript">
    //
    // RP-discovery information
    //
    var txtWarningReturnTo = "WARNING: The return_to URL for the site has not
been successfully validated using relying-party discovery";
    var returntoValidated = @OPENID_RETURN_TO_VALIDATED@;
    if (!returntoValidated) {
        document.write(txtWarningReturnTo);
    }
</script>


    <p />
    Your identity URL is: <b>@OPENID_IDENTITYURL@</b>
<script type="text/javascript">
    //
    // Display claimed identifier if different from identity URL
(e.g. if delegation was being used)
    //
    var txtClaimedID = "Your claimed identifier is: ";
    var identityurl = "@OPENID_IDENTITYURL@";
    var claimedid = "@OPENID_CLAIMEDID@";
    if (claimedid != identityurl) {
        document.write("<p />");
        document.write(txtClaimedID);
        document.write("<b>");
        document.write(claimedid);
        document.write("</b>");
    }
</script>


    <p />
<script type="text/javascript">
    //
    // PAPE information
    //
    var txtMaxAuthnAge = "Requested Maximum Authentication Age (seconds): ";
    var txtRequestedAuthnPolicies = "Requested Authentication Policies";
    var txtRequestedAssuranceLevels = "Requested Assurance Levels";

    var nopii = false;
    var maxAuthenticationAge = @MAXIMUM_AUTHENTICATION_AGE@;
    if ( maxAuthenticationAge >= 0) {
        document.write("<p/>" + txtMaxAuthnAge + maxAuthenticationAge);
    }

    var strAuthPolicies = "";
    [RPT authenticationPolicies]
        strAuthPolicies += "@REQUESTED_AUTHENTICATION_POLICY@"+",";
    [ERPT authenticationPolicies]
    if (strAuthPolicies.length > 0) {
        // strip last comma and split into array
        strAuthPolicies =
strAuthPolicies.substring(0,strAuthPolicies.lastIndexOf(","));
        var authPolicies = strAuthPolicies.split(",");
        document.write("<p/>");
        document.write("<table border>");
        document.write("<tr><th>" + txtRequestedAuthnPolicies + "</th></tr>");
        for (var i = 0; i < authPolicies.length; i++) {
            document.write("<tr><td>"+authPolicies[i]+"</td></tr>");

            // check if this is the nopii policy
            if (authPolicies[i] ==
"http://www.idmanagement.gov/schema/2009/05/icam/no-pii.pdf") {
                nopii = true;
            }
        }
        document.write("</table>");
    }

    var strAssuranceLevels = "";
    [RPT assuranceLevels]
        strAssuranceLevels += "@REQUESTED_ASSURANCE_LEVEL@"+",";
    [ERPT assuranceLevels]

    if (strAssuranceLevels.length > 0) {
        // strip last comma and split into array
        strAssuranceLevels =
strAssuranceLevels.substring(0,strAssuranceLevels.lastIndexOf(","));
        var assuranceLevels = strAssuranceLevels.split(",");
        document.write("<p/>");
        document.write("<table border>");
        document.write("<tr><th>" + txtRequestedAssuranceLevels + "</th></tr>");
        for (var i = 0; i < assuranceLevels.length; i++) {
            document.write("<tr><td>"+assuranceLevels[i]+"</td></tr>");
        }
        document.write("</table>");
    }

</script>


    <form action="@OPENID_SSOURL@" method="post">
      <input type="hidden" name="openid.mode" value="consent_to_authenticate" />
      <div id="DIV_ATTRIBUTES" name="DIV_ATTRIBUTES" style="display: none;">
        The following required attributes have been requested:<br />
        <ul>
        [RPT requiredAttrs]
            <li>@REQUIRED_ATTRIBUTE@</li>
        [ERPT requiredAttrs]
        </ul>
        <p />
        The following optional attributes have been requested. Please select
which attributes you are prepared to send, or select
"All Optional Attributes":<br /><br />
            <input id="chk_all_optional_attributes" type="checkbox"
checked="checked" name="all_optional_attributes"
onClick="allOptionalAttributes()" />
            <label for="chk_all_optional_attributes">
All Optional Attributes</label><br /><br />
        [RPT optionalAttrs]
                <input id="chk_@OPTIONAL_ATTRIBUTE@" type="checkbox"
name="optattr_@OPTIONAL_ATTRIBUTE@" onClick="oneOptionalAttribute()" />
                <label for="chk_@OPTIONAL_ATTRIBUTE@"
>@OPTIONAL_ATTRIBUTE@</label><br />
        [ERPT optionalAttrs]
      </div>
      <p />
      Do you wish to authenticate to this site, sending all
required attributes and
the selected optional attributes?
      <div>
        <input id="rd_permit_forever" type="radio"
name="consent" value="permit_forever"
```

```
checked="checked" /><label for="rd_permit_forever">
Allow Authentication forever
(add to my trusted sites)</label><br/>
        <input id="rd_permit_once" type="radio"
name="consent" value="permit_once" />
<label for="rd_permit_once">Allow Authentication this time only</label><br />
        <input id="rd_deny_once" type="radio" name="consent"
value="deny_once" />
<label for="rd_deny_once">Do not authenticate to this
site this time only</label><br />
        <input id="rd_deny_forever" type="radio" name="consent"
value="deny_forever" />
<label for="rd_deny_forever">Do not ever authenticate to this site
(add to my untrusted sites)</label><br />
      </div>
      <p /><label for="tx_userdata">User data or persona information:</label>
<input id="tx_userdata" type="text" name="userdata" />
      <p /><input type="submit" name="submit" value="Submit" />
    </form>
<script type="text/javascript">
  //
  // if the nopii policy was requested, leave the attribute information hidden
(as we shouldn't send it), otherwise show it
  //
  if (!nopii) {
    showDiv(document.getElementById("DIV_ATTRIBUTES"));
  }
</script>

  </body>
</html>
```

Template HTML page for trusted site management

This page is used at the identity provider. The HTML page is used to manage the persisted set of trusted or untrusted sites. The user establishes the sites through the consent.html page during single sign-on operations.

OpenID identity provider functionality includes the ability to store and retrieve certain user preference attributes including:

- Whether or not a particular consumer site, as identified by `trust_root` value, is trusted. The trust values can be `once`, `never`, or `always`.
- The list of optional attributes that can be sent to a particular trusted consumer
- Any optional user preference data that the identity provider might choose to use when building an attribute set for a single sign-on request to a consumer. For example, the optional detail could include a persona index.

  Tivoli Federated Identity Manager provides a mechanism that stores the attributes in persistent cookies on the browser.

The Tivoli Federated Identity Manager server includes a page template and supporting code. The page template and supporting code use the interface for storing and retrieving information about trusted consumers. Users can use the page template to display and manage this list.

The template file is sitemanager.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The template has the following replacement macros:

@USERNAME@
This macro is replaced with the Tivoli Federated Identity Manager user name.

@SITE_NAME@
This macro is a multi-valued and is used inside either a `[RPT trustedSites]` or `[RPT untrustedSites]` repeatable replacement list. The macro is used to display information about sites that are configured for one of the following states:

- trusted forever
- denied forever

This macro displays the `trust_root` URL for the trusted or untrusted site.

@REQUIRED_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list. The macro is used to display a comma-separated list of the specific set of required attributes that the user must send to the consumer.

@OPENID_SITEMANAGERURL@
This macro is replaced with the URL endpoint of the site manager delegate which is used to process the `remove` action on trusted sites.

@ALL_OPTIONAL_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list to for the trusted site. The macro us used to indicate if the user is prepared to send all requested optional attributes to that consumer. Supported values are `true` or `false`.

@LISTED_OPTIONAL_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list. The macro is used to display a comma-separated list of the specific set of optional attributes the user is prepared to send to that consumer. This value is a non-empty string when `@ALL_OPTIONAL_ATTRIBUTES@` is `false` for the trusted site. When `@ALL_OPTIONAL_ATTRIBUTES@` is `true`, this value is an empty string.

@USERDATA@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list. The macro is used to display optional user data. The data can be specified by a user when the user processed the consent-to-authenticate page, as part of choosing to permanently trust the site. When no user data is specified, the value of the macro is the empty string

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
  <title>OpenID Site Manager</title>
 </head>
 <body>
  OpenID Site Manager</titleb>@USERNAME@</b>
  <p/>
  Trusted Sites<br/>
  <table border="1">
   <tr><td>Site</td><td>Required Attributes</td><td>All Optional
Attributes?</td><td>Permitted Optional Attributes</td><td>User
Data</td><td>Action</td></tr>
[RPT trustedSites]
   <tr>
    <td>@SITE_NAME@</td>
    <td>@REQUIRED_ATTRIBUTES@</td>
    <td>@ALL_OPTIONAL_ATTRIBUTES@</td>
    <td>@LISTED_OPTIONAL_ATTRIBUTES@</td>
    <td>@USERDATA@</td>
    <td><a href="@OPENID_SITEMANAGERURL@?action=
remove&site=@SITE_NAME@">Remove</a></td>
   </tr>
 [ERPT trustedSites]
 </table>
 <p/>
 Untrusted Sites<br/>
 <table border="1">
 <tr><td>Site</td><td>Action</td></tr>
 [RPT untrustedSites]
 <tr>
  <td>@SITE_NAME@</td>
  <td>@OPENID_SITEMANAGERURL@?
action=remove&site=@SITE_NAME@">Remove</a></td>
 </tr>
[ERPT untrustedSites]
 </table>
</body>
</html>
```

Figure 34. Template HTML file sitemanager.html

This template is used on the identity provider only.

Template page for OpenID error

When an error halts processing on the identity provider or the consumer, and the error is not returned to, Tivoli Federated Identity Manager uses a generic error page template to display detailed error text information.

For example:

- On an identity provider this page is used when processing the trusted sites page or when a single sign-on request lacks a valid `return_to` URL.
- On a consumer, this page is used when bad parameters are returned in the login page.

The template page is error.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The following replacement macros are supported:

@REQ_ADDR@
: This macro is replaced with the URL of the currently called delegate endpoint.

@TIMESTAMP@
: This macro is replaced with the current time in UTC.

@DETAIL@
: This macro is replaced with the native language support (NLS) text of the error message associated with the error.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of any exception that caused the error.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>An OpenID error has occurred</title>
    </head>
    <body style="background-color:#ffffff">
        <div>
            <h2 style="color:#ff8800">An error has occurred</h2>
            <div id="infoDiv" style="background-color:#ffffff;color:#000000">
                <em>@REQ_ADDR@</em> <br />
                <em>@TIMESTAMP@</em> <br />
            </div>
            <br />
<div id="detailDiv" style="background-color:#999999; border-style:solid;
border-width:1px; border-color:#000000">
                <h4>Error details</h4>
                @DETAIL@
            </div>
            <br />
            <div id="stackDiv" style="background-color:#999999;
border-style:solid; border-width:1px; border-color:#000000">
                <h4>Stack trace</h4>
                @EXCEPTION_STACK@
            </div>
        </div>
    </body>
</html>
```

Figure 35. Template HTML file error.html

This template is used on both the identity provider and consumer.

Template page for OpenID 2.0 indirect post

OpenID 2.0 specifies that HTTP POST requests can be used instead of HTTP redirects, to send indirect messages between the identity provider and relying party (consumer). The messages are sent to the browser and then redirected to the target.

Tivoli Federated Identity Manager automatically switches messages into a self-posting FORM using HTTP POST (rather than a 302 redirect) when the following conditions are true:

- OpenID 2.0 is being used
- The message size exceeds 2K bytes

When POST is used, a page is loaded with a self-posting FORM (rather than a 302 redirect) containing the same parameters that would otherwise have been passed on the query string.

The template file is indirect_post.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The file supports the following replacement macros:

@OPENID_PARTNER_URL@
: This macro is replaced with the URL of the destination partner. This is used for the FORM action parameter.

@PARAM_NAME@ / @PARAM_VALUE@
: These are multi-valued macros that are used inside a `[RPT formFields]` repeatable replacement list. They are used for parameters to pass to the recipient.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>OpenID Message</title>
  </head>
  <body>
    <form method="post" name="openid_message" action="@OPENID_PARTNER_URL@">
      [RPT formFields]
        <input type="hidden" name="@PARAM_NAME@" value="@PARAM_VALUE@" />
      [ERPT formFields]
      <noscript>
      <button type="submit">Send OpenID Message</button>
<!-- included for requestors that do not support javascript -->
      </noscript>
    </form>
    <script type="text/javascript">
        var signOnText = 'Sending OpenID message...';
        document.write(signOnText);
        setTimeout('document.forms[0].submit()', 0);
    </script>
  </body>
</html>
```

Figure 36. Template file indirect_post.html

This template is used on the identity provider only.

Template page returned for checkid_immediate

When a checkid_immediate request is initiated by the Tivoli Federated Identity Manager service provider, and the identity provider returns a status that it cannot determine whether the user owns the URL, the identity provider also returns one of the following:

- openid.user_setup_url

  For OpenID 1.1
- openid.mode=user_setup_needed

  For OpenID 2.0

When the Tivoli Federated Identity Manager consumer receives this type of reply, it returns a page template file.

The page template file is immediate.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The file has the following replacement macro:

@OPENID_USER_SETUP_URL@

This macro is replaced with the URL returned in the openid.user_setup_url parameter of an identity provider response to the checkid_immediate request. When the request is an OpenID 2.0 request, this parameter can be the empty string.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>Results from checkid_immediate</title>
  </head>
  <body>
    <script type="text/javascript">
        var setup_url = "@OPENID_USER_SETUP_URL@";
        if (setup_url) {
            document.write('<a href="');
            document.write(setup_url);
            document.write('">Please click here to complete identity
            provider requirements</a>');
        } else {
            document.write('Unable to proceed as authentication is required at
        the OpenID Identity Provider.');
        }
    </script>
  </body>
</html>
```

Figure 37. Template page immediate.html

This template is used on the consumer only.

Template page returned for server error

When the Tivoli Federated Identity Manager consumer sends a checkid_immediate or checkid_setup request, and the request results in an error, the identity provider server returns `openid.mode` set to `error` and specifies the error text in `openid.error`.

In this case, the consumer returns the server_error.html page.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The template page supports the following replacement macros:

@OPENID_SERVER@
This macro is replaced with the OpenID server URL that the consumer was communicating with when the error occurred.

@OPENID_ERROR@
The text from `openid.error`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>OpenID Error Returned By Server</title>
  </head>
  <body>
    The OpenID Server: @OPENID_SERVER@ has returned
    the following error text:<br />
    @OPENID_ERROR@
  </body>
</html>
```

Figure 38. Template file server_error.html

This template is used on the consumer only.

Chapter 24. Planning a Liberty federation

You will need to specify values for federation properties when configuring a Liberty federation.

You should be familiar with the Liberty standards documents before implementing a single sign-on federation. The standards specify data exchange and message processing. You should understand what information you are required to provide to your business partners, and what information your partner must provide to you.

Liberty Alliance

`http://www.projectliberty.org`

The Federation wizard will prompt you to supply values for a number of properties. Most of them can be modified later, after federation creation.

The choice of profile (or profiles) to use is based on both business policy decisions and network security architecture. Federation partners must agree on the profile choices in order to enable user single sign-on across the federation. The choice must be made prior to configuring the federation.

The Liberty standard supports a unique range of single sign-on profiles. The profiles extend beyond specifications for achieving federated single sign-on, and can include other functions such as single logout, federation termination notification and register name identification.

Identity provider and service provider roles

Each partner in a federation has a role. The role is either **Identity Provider** or **Service Provider**.

- Identity provider

  An identity provider is a federation partner that vouches for the identity of a user. The Identity Provider authenticates the user, and provides an authentication token to the service provider.

  The identity provider either directly authenticates the user, such as by validating a user name and password, or by indirectly authenticating the user, by validating an assertion about the user's identity, as presented by a separate identity provider.

  The identity provider handles the management of user identities in order to free the service provider from this responsibility.

- Service Provider

  A service provider is a federation partner that provides services to end user. Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on identity providers to assert the identity of a user, and rely on identity providers to manage user identities for the federation.

  Service providers can maintain a local account for the user, which can be referenced by an identifier for the user.

Liberty single sign-on profiles

Liberty supports more than one single sign-on profile. You must select at least one profile. You can optionally configure both Browser artifact and Browser POST profiles when configuring an identity provider. You can configure only one profile when configuring a service provider.

Browser artifact

Browser artifact uses a SOAP backchannel to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

You can optionally configure browser artifact when configuring an identity provider or a service provider.

When you select browser artifact, you will also be prompted to enter the name of an encryption key for the trusted session. You must specify a key even if you choose to not require the signing of assertions for other Liberty message communications.

Browser POST

Browser POST uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

You can optionally configure browser POST when configuring an identity provider or a service provider.

Note: When configuring an identity provider, you can select both browser artifact and browser POST profiles. However, when configuring a service provider, you can select only one profile – either browser artifact or browser POST.

Liberty-enabled client/proxy (LECP) single sign-on profile

A Liberty-enabled client or Liberty-enabled proxy has, or knows how to obtain, the information that is required to be able to connect to the identity provider that the user (principal) wants to use with the service provider. A Liberty-enabled proxy is an HTTP proxy such as a Wireless Application Protocol (WAP) gateway that emulates a Liberty-enabled client.

LECP Providers

A comma delimited list of header variables used by LECP. This property is set when configuring both identity providers and service providers. There is no default value.

An example of single header variable:

```
ibm_msisdn
```

An example of multiple header variables:

```
ibm_msisdn,x_msisdn
```

Liberty register name identifier

This profile updates the identifier for a specific user or principal. Liberty requires identity providers and service providers to exchange an alias (also called an identifier) to each user account, instead of exchanging the user's real account name. This ability enables account linkage while hiding the user account name.

Configuration of the Register Name Identifier profile is optional.

When the profile is selected, the administrator must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

  The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Updates to name identifiers are accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers.
- SOAP/HTTP

  Updates to name identifiers are accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Register Name Identifier Service URL
The URL endpoint is used for user-agent-based Register Name Identifier protocols. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/rni
```

Register Name Identifier Return URL
The URL endpoint used for redirection after HTTP name registration has taken place. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/rnireturn
```

This value is required for RNI with HTTP Redirect communication. This value is not required for SOAP/HTTP communication.

Liberty federation termination notification

This profile terminates account linkage across the federation for a specified user. This profile is disabled by default.

Configuration of this profile is optional. When the profile is selected, you must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

  The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Termination of account federation is accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers.
- SOAP/HTTP

  Termination of account federation is accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Federation Termination Notification Service URL
The URL on the provider to which single federation termination notification processes are sent. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/ftn
```

Federation Termination Notification Return URL

The URL used by the identity or service Provider when redirecting the user agent at the end of the user-agent-based federation termination notification process.

```
https://idp.example.com/FIM/sps/libertyfed/liberty/ftnreturn
```

This value is required for FTN when using HTTP Redirect communication.

Liberty single logout

This profile terminates all login sessions within the federation for a specified user. This profile is disabled by default.

Configuration of this profile is optional. When the profile is selected, the administrator must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

  The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Logout of user sessions is accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers
- HTTP GET

  Identity providers can use Image Tags to cause the browser to use HTTP GET to communicate the logout requests to the service providers. Logout requests are processed concurrently rather than serially. If a logout request fails, any remaining logout requests are unaffected, and are sent to the appropriate service provider. By contrast, when logout requests are processed serially (HTTP redirect) a failed logout request cancels any remaining logout requests.

  Note: This option is specified only on identity providers. Service providers cannot set this option.
- SOAP/HTTP

  Logout of user sessions is accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Single Logout Service URL

The URL to which the service provider sends a request to log out a user. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/slo
```

Single Logout Return URL

The URL used by the service provider when redirecting the user agent to the identity provider at the end of the single logout profile process. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/sloreturn
```

This value is required for SLO using HTTP Redirect communication.

Liberty identity provider introduction

Identity Provider Introduction enables a service provider to discover which identity providers are used by a user (Principal). The Introduction profile relies on a cookie that is written in a domain that is common between identity providers and service providers in an identity federation network.

This profile is configured only on an identity provider.

Common DNS Domain

The common DNS domain is a virtual domain into which a component is configured to set or retrieve a cookie. Use of this common domain enables identity providers and service providers, which typically exist in separate domains, to access a cookie. The domain does not have to exist prior to setting this configuration property. However, you must create it before a user attempts single sign-on while relying on the identity provider introduction profile. This property is set only when configuring an identity provider. There is no default value. For example:

```
cot.projectliberty.org
```

IPI configuration requires that this field contain a value.

Common Domain Hostname

The name of a host system in the common DNS domain. This host receives requests to either set or read the common domain cookie used by the identity provider introduction profile. This property is set only when configuring an identity provider. There is no default value. For example:

```
idp.cot.projectliberty.org
```

The domain name portion of this host name must match the value specified in the Common DNS Domain. In this example, the host system value must include `cot.projectliberty.org`.

IPI configuration requires that this field contain a value.

Liberty message security

Digital signature options

The federation creation wizard asks you whether you want to sign Liberty messages. When you chose to sign Liberty messages, you must specify a key or certificate to use.

In some cases, when you do not select Sign Liberty Messages, you must still enter a key or certificate. For example:

- When you select browser artifact profile, you must specify a key to be used to sign messages that are sent across the backchannel for the artifact.
- When you select one of the optional profiles, and also specify SOAP communication to be initiated by the service provider, you must specify a key or certificate.

When you need to enter a key or certificate, you must provide the following configuration information:

Keystore file name

The wizard presents a choice of the keystores that you configured before you began configuration of the single sign-on federation.

Keystore password
You must supply the password for the keystore you specify.

Key name
You must specify which key to use.

Liberty communication properties

Liberty Message Lifetime

An integer value indicating the amount of time, in seconds, that a Liberty message remains valid. This property is set on both the identity provider and the service provider.

Minimum value: 60 seconds

Maximum: No maximum other than the maximum integer supported by the data type.

Default: 60 seconds.

Liberty Artifact Lifetime

An integer indicating the time, in seconds, in which a service provider must retrieve an assertion from an identity provider. The service provider uses an artifact to retrieve the assertion. The identity provider keeps the mapping of the artifact to the assertion in its cache for this amount of time. When the service provider does not collect the artifact in this amount of time, the cache purges the artifact and the service provider login fails.

This property is specified only when configuring an identity provider with browser artifact single sign-on profile.

Note: This value is not used for browser POST profile.

Minimum value: 120 seconds

Default: 120 seconds.

Require Consent to Federate

Enables or disables the requirement that the identity provider prompt the end user to consent to joining the federation. This property is set on the identity provider only. This message is presented when federating of the user account occurs. Default value is disabled. Select the check box to enable issuing of the prompt.

SOAP Endpoint

The Simple Object Access Protocol (SOAP) endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

This setting is required when one or both of the following conditions is true:

- Browser artifact single sign-on profile is selected on the Liberty profiles window.
- One or more of the optional Liberty profiles are selected and SOAP/HTTP communication initiated by at least one of the service providers is selected.

For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/soap
```

Single Sign-on is Passive (Identity Provider does not interact with user)

Enables or disables the requirement that the identity provider must not interact with principal (user) and must not take control of the user interface from the service provider. This property is set only when configuring a service provider. Select the check box to enable this requirement. Default value: Disabled

Force Identity Provider to authenticate user

Enables or disables a requirement that the identity provider must authenticate a user (Principal) regardless of whether the user is already authenticated. This value is specified only when the **Single Sign-on is Passive (Identity Provider does not interact with user)** check box is cleared. This property is set only when configuring a service provider.

When this setting is cleared, the identity provider must authenticate the user (Principal) only when the user is not presently authenticated.

- Select the **Force Identity Provider to authenticate user** check box to enable this requirement.
- Clear the check box to disable this requirement.

Liberty token modules

When you create a single sign-on federation, you must configure an instance of a security token module for the federation. The token module corresponds to a security token type that defines the format for the encrypted token that contains user credential information.

The token is exchanged between the identity provider and service provider as part of the authentication and authorization services for the processing of each user access request.

When you use the federation creation wizard, a token type is automatically selected for you based on your choice of single sign-on protocol.

Configuration of the Liberty token module is required only on the identity provider. No configuration is required when deploying a service provider.

The configuration property is the same for both Liberty v1.1 tokens and Liberty v1.2 tokens.

Amount of time the assertion is valid after being issued (seconds)

An integer value that specifies the number of seconds that the assertion remains valid. This is specified for Liberty tokens. The minimum value is 120 seconds. The maximum value is 300 seconds.

Liberty identity mapping

The federation creation wizard will prompt you to specify either an XSLT mapping rule file or a custom mapping module instance.

The XSLT mapping file or custom mapping module instance must be prepared before you configure the federation.

XSLT Transformation for Identity Mapping

Selection of this button in the wizard indicates that you will provide an XSL file containing the identity mapping. Enter the name of a file on the local file system.

Custom Mapping Module Instance
Selection of this button in the wizard indicates that you will provide a custom mapping module instance to use instead of an XSL file. You will be prompted to enter any configuration properties that your custom mapping module instance requires.

Mapping a Tivoli Access Manager credential to a Liberty or SAML 2 token

This scenario occurs when messages are exchanged between partners in a Liberty or SAML 2 single sign-on federation, and user identity information is managed by Tivoli Access Manager. When a user request is received (for example, for access to a remote resource) the trust service contacts Tivoli Access Manager and obtains a Tivoli Access Manager credential for the user identity.

In this scenario, the trust service Tivoli Access Manager credential module operates in validate mode. In this mode, it converts the Tivoli Access Manager credential to an Input STS universal user document (In-STSUUSER). The In-STSUUSER that is created from the Tivoli Access Manager credential module has all of the information from the credential. This information is available for possible use by the trust service module that will build the outgoing token.

The trust service consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created.

Next, the identity mapping module converts the In-STSUUSER into an Output STS universal user (Out-STSUUSER). The Out-STSUUSER must contain the information that is needed by the Tivoli Federated Identity Manager Liberty (or SAML 2) token module to generate a Liberty (or SAML 2) token.

The Out-STSUUSER must contain the following information in order for the token module to be able to generate a valid token:

Table 102. Out-STSUUSER entries used to generate a Liberty or SAML 2 token

Out-STSUUSER element	Token Information	Required?
Principal Attr: Name	Name to be passed to the alias service	Required
Attribute: AuthenticationMethod	The authentication method. Note that this element is always set to "password" (Username/password) regardless of the authentication mechanism set in the Tivoli Access Manager credential.	Required
Attribute List	Additional custom attributes.	Optional

The mapping module is responsible for:

1. Mapping Principal Attr Name in In-STSUUSER to a Principal name entry in the Out-STSUUSER.

   Note that when the token module generates the token, this Principal name is not directly used. Instead, the value in the Name field is sent as input to the Tivoli Federated Identity Manager alias service. The alias service obtains the alias (name identifier) for the principal, and places the returned alias in the generated token module.

Figure 39 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl. Note that Liberty tokens are extensions to SAML tokens. Therefore, comments in the sample code referring to SAML tokens are correct in this context.

```
</xsl:template>
<!-- This template replaces the entire Principal element with one that contains
     just the email address (from the ivcred tagvalue_email) and the data type
     appropriate for SAML. -->
 <xsl:template match="//stsuuser:Principal">
  <stsuuser:Principal>
   <stsuuser:Attribute name="name"
                 type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
    <stsuuser:Value>
    <xsl:value-of
    select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_email']
          [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:Principal>
 </xsl:template><!--
```

Figure 39. XSL code sample showing mapping of a value from a Tivoli Access Manager credential into a Principal name for a Liberty token

2. Setting the authentication method to the "password" mechanism, regardless of the value obtained from the Tivoli Access Manager credential. This action is required by the token module.

   Figure 40 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList><!-- First the authentcation method attribute -->
   <stsuuser:Attribute name="AuthenticationMethod"
                           type="urn:oasis:names:tc:SAML:1.0:assertion">
    <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
   </stsuuser:Attribute>
          ....
       </stsuuser:AttributeList>
</xsl:template>
```

Figure 40. XSL code sample showing assignment of authentication method as an Attribute for a Liberty token.

3. Populating the attribute statement of the assertion with the attributes in the AttributeList in the In-STSUUSER. This information becomes custom information in the token.

   There can be custom attributes that are required by applications that will make use of information that is to be transmitted between federation partners.

   Figure 41 on page 318 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
       <!-- Now the commonName attribute -->
   <stsuuser:Attribute name="commonName"
                           type="http://example.com/federation/v1/commonName">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_name']
           [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
   <!-- Now the ssn attribute -->
   <stsuuser:Attribute name="ssn" type="http://example.com/federation/v1/ssn">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_ssn']
             [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 41. XSL code sample showing assignment of optional attributes for a Liberty token

4. Note that the GroupList element of the In-STSUUSER is not read by the token module. However, information in this element can optionally be used to populate custom attributes of the Out-STSUUSER.

   Figure 42 shows the optional assignment of a GroupList value to an attribute. This code sample is from the demonstration application mapping file ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
       <!-- Now the role attribute (can be multi-valued) -->
   <stsuuser:Attribute name="role"
                           type="http://example.com/federation/v1/role">
    <xsl:for-each select="//stsuuser:GroupList/stsuuser:Group">
     <stsuuser:Value>
      <xsl:value-of select="@name" />
     </stsuuser:Value>
    </xsl:for-each>
   </stsuuser:Attribute>
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 42. XSL code sample showing optional assignment of GroupList value to an attribute for a Liberty token

Mapping a Liberty or SAML 2 token to a Tivoli Access Manager credential

The service provider receives a Liberty or SAML 2 token. The token module, operating in validate mode, creates an In-STSUUSER document from the token. Table 103 on page 319 shows the information from the token that is converted into an In-STSUUSER document.

Table 103. Token information that is converted into a STS universal user document

Token Information	In-STSUUSER element	Required in Out-STSUUSER?
UserID obtained from the alias service	Principal Attr: Name	Required
Additional custom attributes	Attribute List	Optional

Note that the token module does not populate the GroupList element in the In-STSUUSER document.

The token module reads the token and obtains the NameIdentifier. The token module passes the NameIdentifier (an alias) to the alias service. The alias service converts the received alias to the local Tivoli Access Manager User ID. The token module puts the User ID into the Principal element in the In-STSUUSER document.

The trust service must convert this information to a Tivoli Access Manager credential, in order to make an authorization decision on the request from the user identity.

- The NameIdentifier alias that is returned is used to populate the `name` attribute of the Principal. This is the local user ID.

  Figure 43 shows the assignment of a set value for the Principal name. This code sample is from the demonstration application mapping file sp_liberty.xsl.

```
 <!-- This will replace the principal name (which was the email address
      in the SAML assertion) with the user "me_guest". -->
    <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
            <stsuuser:Attribute name="name"
                                type="urn:ibm:names:ITFIM:5.1:accessmanager">
             <stsuuser:Value>
<xsl:value-of
 select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value" />
             </stsuuser:Value>
           </stsuuser:Attribute>
        </xsl:template>
```

Figure 43. XSL code sample showing assignment of a value for the Principal name for a Liberty token.

- Other information from the token is used to populate Attributes in the Attribute List.

  Figure 44 on page 320 shows the optional assignment of additional values to attributes. This code sample is from the demonstration application mapping file sp_liberty.xsl

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
              ....
      <!-- The tagvalue_sso attribute -->
      <stsuuser:Attribute name="tagvalue_sso"
                          type="urn:ibm:names:ITFIM:5.1:accessmanager">
                   <stsuuser:Value>isSingleSignOn</stsuuser:Value>
      </stsuuser:Attribute>
      <!-- The tagvalue_fedname attribute -->
      <stsuuser:Attribute name="tagvalue_fedname"
                         type="urn:ibm:names:ITFIM:5.1:accessmanager">
                    <stsuuser:Value>libertyfed</stsuuser:Value>
      </stsuuser:Attribute>
              ....
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 44. XSL code sample showing optional assignment of attributes for a Liberty token.

Liberty alias service

The Liberty standards for single sign-on protocols standards require the use of aliases when a user identity is sent in a message between partners in a single sign-on federation. The standards require aliases as a method of increasing the privacy of the end user when accessing resources at a service provider.

The specifications refer to the aliases as *Name Identifiers*. Name identifiers for a user are registered during account federation (account linkage) and are thereafter used in all messages between partners. Aliases are randomly generated and do not contain any meaningful identity information.

For each user, a different Name Identifier is required for use with each partner. Optionally, a different Name Identifier can be created for messages in each direction. This capability means that a different alias is used for a user when the identity provider contacts the service provider rather than when the service provider sends a message to the identity provider.

Tivoli Federated Identity Manager provides an alias service that handles the alias management tasks. This service hides most of the alias generation and exchange tasks from the federation administrator. The alias service provides the following services:

- Generation of new aliases and association of them with local users
- Look up of a local user identity when an alias is received from a partner
- Look up of the alias for a local user when the provider needs to send a message to a partner

The Tivoli Federated Identity Manager alias service stores alias information in a user registry. The alias service supports the following user registries:

- IBM Tivoli Directory Server
- Sun ONE

For each of these LDAP servers, you will set some configuration parameters after you have created the Liberty federation.

The alias service does not support Lotus Domino or Microsoft Active Directory user registries. You can write your own alias service for use with those registries.

Chapter 25. Configuring a Liberty federation

About this task

Complete the following tasks:

Procedure

1. "Creating a Liberty identity provider"
2. "Creating a Liberty service provider" on page 325
3. "Configuring a WebSEAL point of contact server for the Liberty federation" on page 327
4. "Exporting Liberty federation properties" on page 328
5. "Exporting SOAP endpoint authentication information to a Liberty partner" on page 328
6. "Obtaining metadata from a Liberty federation partner" on page 329
7. "Importing SOAP endpoint authentication information from a Liberty partner" on page 330
8. "Adding a partner to a Liberty federation" on page 332

Creating a Liberty identity provider

Procedure

1. Log in to the management console and click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets are displayed. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The General Information panel is displayed.
3. Enter a name for the federation and select a role. Click **Next**.
4. Enter the contact information and click **Next**.
5. Select the Liberty 1.1 or Liberty 1.2 protocol and click **Next**. The Point of Contact Server panel is displayed.
6. Enter the point of contact address and click **Next**.
7. Specify the profiles to use with this federation. When finished, click **Next**.
   a. Select at least one of the Liberty single sign-on profiles.

      Liberty supports three single sign-on profiles. You must select at least one profile. You can optionally select both Browser artifact and Browser POST profiles when configuring an identity provider. You can select only one profile when configuring a service provider.

   b. Select any of the optional profiles that you want to configure:
      - **Register Name Identifier**
      - **Federation Termination Notification**
      - **Single Logout**
      - **Identity Provider Introduction**.

        For identity providers only.

8. The Digital Signature Options panel is displayed. Select or clear the check box for **Sign Liberty Messages**. When you chose to sign Liberty messages, you must specify a key or certificate to use.

   In some cases, when you do not select Sign Liberty Messages, you must still enter a key or certificate. For example:

   - When you select browser artifact profile, you must specify a key to be used to sign messages that are sent across the backchannel for the artifact.
   - When you select one of the optional profiles, and also specify SOAP communication to be initiated by the service provider, you must specify a key or certificate.
9. When you need to enter a key or certificate, select a keystore and enter the keystore password. Click **List Keys** to display the keys or certificates contained in the selected keystore. Select a key and click **Next**
   - The password for the default **DefaultKeyStore** keystore is `testonly`.
   - A sample key is provided for test purposes only. Do not use this key in a production environment.
10. Configure the Liberty data properties:
    a. A default value is provided for the **SOAP endpoint**. Use this value unless there is an endpoint conflict on your host.
    b. Specify **Liberty Message Lifetime**.
    c. Specify **Liberty Artifact Lifetime**.
    d. Select or clear the check box for **Require Consent to Federate**.
    e. When LECP profile has been selected, enter **LECP providers**.
    f. When Identity Provider Introduction has been selected, enter the **Common DNS Domain** and **Common Domain Hostname**.
    g. Click **Next**.

    The Liberty Token Module Configuration panel is displayed. The panel contents are the same for both Liberty v1.1 tokens and Liberty v1.2 tokens.
11. Specify a value in the **Amount of time the assertion is valid after being issued (seconds)** field and click **Next**.
12. The Identity Mapping Options panel is displayed. Select one of the radio buttons.
    - Use XSL Transformation for Identity Mapping

      Indicates that you will provide an XSL file containing the required identity mapping.

      a. When you select this choice and click **Next**, the Identity Mapping panel is displayed. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

         This is a file that you have prepared prior to this installation.

         Optionally you can click the **Browse** button to locate the file on the local file system.

      b. Click **Next**.

         An error is displayed if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).
    - Use Custom Mapping Module Instance

      Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

a. When you select Use Custom Mapping Module Instance, a table of Module Instances is displayed. Select the radio button for the module instance to use and click **Next**.

b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel displays a message indicating that there are no properties to configure for the specified module instance.

13. The Summary panel is displayed. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See:

- "Configuring a WebSEAL point of contact server for the Liberty federation" on page 327

Creating a Liberty service provider

Procedure

1. Log in to the management console and click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets are displayed. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The General Information panel is displayed.
3. Enter a name for the federation and select a role. Click **Next**.
4. Enter the contact information and click **Next**.
5. Select the Liberty 1.1 or Liberty 1.2 protocol and click **Next**. The Point of Contact Server panel is displayed.
6. Enter the point of contact address and click **Next**.
7. Specify the profiles to use with this federation. When finished, click **Next**.

   a. Select at least one of the Liberty single sign-on profiles.

      Liberty supports three single sign-on profiles. You must select at least one profile. You can optionally select both Browser artifact and Browser POST profiles when configuring an identity provider. You can select only one profile when configuring a service provider.

   b. Select any of the optional profiles that you want to configure:

      - **Register Name Identifier**
      - **Federation Termination Notification**
      - **Single Logout**
      - **Identity Provider Introduction**.

        For identity providers only.

8. The Digital Signature Options panel is displayed. Select or clear the check box for **Sign Liberty Messages**. When you chose to sign Liberty messages, you must specify a key or certificate to use.

   In some cases, when you do not select Sign Liberty Messages, you must still enter a key or certificate. For example:

   - When you select browser artifact profile, you must specify a key to be used to sign messages that are sent across the backchannel for the artifact.

- When you select one of the optional profiles, and also specify SOAP communication to be initiated by the service provider, you must specify a key or certificate.

9. When you need to enter a key or certificate, select a keystore and enter the keystore password. Click **List Keys** to display the keys or certificates contained in the selected keystore. Select a key and click **Next**
   - The password for the default **DefaultKeyStore** keystore is `testonly`.
   - A sample key is provided for test purposes only. Do not use this key in a production environment.
10. Configure the settings for Liberty profile service provider:

    Note: For information on each of the properties on this panel, see .

    a. If you selected an optional Liberty profile (register name identifer, federation termination notification or single logout), and you chose SOAP/HTTP as the communication protocol, you must specify a SOAP endpoint. A default value is provided. You can accept the default unless you have a specific configuration requirement that calls for a different SOAP endpoint.

    b. Specify a value in the **Liberty Message Lifetime (in seconds)** field.

    c. Select or clear the check mark for **Single Sign-on is Passive (Identity Provider does not interact with user)**

    d. Select or clear the check mark for **Force Identity Provider to authenticate user**.

    e. If you selected LECP single sign-on profile, enter a **LECP Provider.** Click **Next**.
11. The Identity Mapping Options panel is displayed. Select one of the radio buttons.
    - Use XSL Transformation for Identity Mapping

      Indicates that you will provide an XSL file containing the required identity mapping.

      a. When you select this choice and click **Next**, the Identity Mapping panel is displayed. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

      This is a file that you have prepared prior to this installation.

      Optionally you can click the **Browse** button to locate the file on the local file system.

      b. Click **Next**.

      An error is displayed if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).
    - Use Custom Mapping Module Instance

      Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

      a. When you select Use Custom Mapping Module Instance, a table of Module Instances is displayed. Select the radio button for the module instance to use and click **Next**.

      b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel displays a message indicating that there are no properties to configure for the specified module instance.

12. The Summary panel is displayed. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.
13. Click **Restart WebSphere**.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See:

- "Configuring a WebSEAL point of contact server for the Liberty federation"

Configuring a WebSEAL point of contact server for the Liberty federation

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the Liberty federation.

Before you begin

The federation wizard provides a button that you can use to obtain a configuration utility.

About this task

You must obtain the utility and run it. Complete the following steps:

Procedure

1. From the management console, click **Download Tivoli Access Manager Configuration Tool**
2. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
3. Return to the management console, and Click **Done** to return to the Federations panel.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation we will complete other tasks first.
4. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -cfgfile webseald-instance_name.conf
    -action tamconfig
   ```

   You will need to know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -cfgfile webseald-default -action tamconfig
```

For more information, see:

- Appendix A, "tfimcfg reference," on page 491

What to do next

The next task is to export your Liberty federation properties to a file. See "Exporting Liberty federation properties."

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log in to the administration console.
2. Click Tivoli Federated Identity Manager > Manage Configuration > Point of Contact
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Exporting Liberty federation properties

About this task

When you want to join a federation hosted by another business partner, you must supply your federation configuration properties. Use the management console to create a metadata file that contains the properties for your federation. Give this file to your federation partner.

Procedure

1. Log in to the management console. Click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**.
2. The Federations panel is displayed. Select your Liberty federation from the table.
3. Click **Export**. The browser displays a message window that prompts you to save the file containing the exported data. Click **OK**. The browser download window prompts for a location to save the file.
4. Select a directory and file name and click **Save**. Place the file in an easily accessible location.

What to do next

You will need to provide this file to your partner, when your partner wants to import configuration information for this federation.

Exporting SOAP endpoint authentication information to a Liberty partner

Supply your partner with any keys, certificates, user names or passwords needed to complete SSL communication over SOAP ports.

Before you begin

Note: The securing of SOAP ports with SSL, and the accompanying use of keys, certificates, user names, and passwords, is not required but is typically done as best practice for optimal network security.

About this task

Liberty provides a SOAP backchannel that is used with browser artifact single sign-on profile, and can be used with additional Liberty profiles that support SOAP binding. The SOAP backchannel can optionally be protected through the use of SSL (HTTPS endpoints). Use of SSL is common for SOAP endpoints.

For Liberty federations, you might also need to provide authentication information (certificates and basic authentication information) to your partner, for use when accessing SOAP endpoints.

This task is done outside of the management console.

Note: If your federation does not use SSL to secure SOAP ports, you can skip this task

Procedure

1. Provide your partner with a validation certificate that the partner will use to validate SSL communication from your federation provider, when messages that your provider has initiated are received at the partner's SOAP endpoint.
2. If you want your partner to authenticate as a client, you must specify whether the partner is to use client certificate authentication or basic authentication. Only one form of authentication can be specified.

   Client certificate authentication

   - When you require client certificate authentication, you and your partner must decide which certificate the partner must present when establishing the SSL session. The choice of certificate is a business decision. The certificate can be one that your partner already has, or one that you provide to your partner for this purpose.

   Basic authentication

   - When you require basic authentication, you must supply your partner with a user name and password to be used when establishing an authenticated session

Obtaining metadata from a Liberty federation partner

When you want to add your business partner as a partner to your Liberty single sign-on federation, you must obtain from them the necessary configuration information about their Liberty federation.

Before you begin

Your partner must have already installed and configured a Liberty federation. The partner's federation plays the opposite role to your federation. For example, when your federation is configured as an identity provider, your partner's federation is configured as a service provider.

About this task

Procedure

1. Your partner must export configuration information about the Liberty federation into a metadata file.

   The partner will use the Tivoli Federated Identity Manager management console to export the configuration settings to the metadata file. This is the same feature that you used to provide your configuration settings to your partner.

   The Export feature uses a naming scheme for metadata files, based on the name of the federation and a time stamp. The administrator can override the default name for the metadata file, and change the name to any arbitrary name.
2. Your partner will provide you with the metadata file.

   This action takes place outside of the Tivoli Federated Identity Manager console. Your partner will use whatever process has been agreed to as part of the business agreement that was previously negotiated between the partner companies.
3. Place the metadata file onto the local file system where the Liberty federation configuration is kept. You can choose any location for the metadata file. You have now completed the preparation task. You will later use the Tivoli Federated Identity Manager management console to add the partner to your Liberty federation. The console provides a Add Partner wizard that prompts you to supply the name of the file containing the partner's metadata. The documentation will guide you through this task at the appropriate time.

Importing SOAP endpoint authentication information from a Liberty partner

About this task

Liberty provides a SOAP backchannel that is used with browser artifact single sign-on profile, and can be used with additional Liberty profiles that support SOAP binding. The SOAP backchannel can optionally be protected through the use of SSL (HTTPS endpoints). Use of SSL is common for SOAP endpoints.

For Liberty federations, you might need to obtain authentication information (certificates and basic authentication information) from your partner, for use when you want to access SOAP endpoints on the partner's federation.

This task is done outside of the management console.

Note: If your partner's federation does not use SSL to secure SOAP ports, you can skip this task

Procedure

1. Obtain from your partner a validation certificate that the you will use to validate SSL communication from your partner, when messages that your partner has initiated are received at your SOAP endpoint.
2. Obtain from your partner the requirement, if any, for your client to authenticate when your client wants to contact your partner's SOAP endpoint.

   If your partner wants your client to authenticate, the partner must tell you whether to use client certificate authentication or basic authentication. Only one form of authentication can be specified.

Client certificate authentication

- When your partner requires client certificate authentication, you and your partner must decide which certificate you will present when establishing the SSL session. The choice of certificate is a business decision. The certificate can be one that you already have, or can be one that your partner gives you to use for this purpose.

Basic authentication

- When your partner requires basic authentication, your partner must supply you with a user name and password to be used when establishing an authenticated session

3. When your partner uses SSL communication for SOAP ports, you must import the certificate you obtained from your partner. You can import the certificate into any keystore that is managed by the Tivoli Federated Identity Manager key service.

   Note: Tivoli Federated Identity Manager provides a default keystore (DefaultTrustedKeystore) that contains some common CA certificates that you might be able to use as validation certificates. In most cases, however, you must import a certificate obtained from your partner.

   a. Click **Tivoli Federated Identity Manager -> Key Service**.

      The Keystores panel is displayed.

   b. Select a Keystore from the Keystore table. The **View Keys** button is activated.

   c. Click **View Keys**. The Keys panel is displayed. Keys in the selected keystore are listed.

   d. Click the **Import** button. The Key Wizard starts and displays the Keystore Format panel.

   e. Select the appropriate **Keystore format** for the file you want to import.

      (PEM)
         (Privacy-Enhanced Message) Public certificate

      PKCS#12
         Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard

   f. For PKCS#12, specify whether the keystore contains multiple key pairs.

      1) Select the **Contains multiple key pairs** field when appropriate.

      2) Clear (remove the check mark from) the **Contains multiple key pairs** field when there is only one key pair. The key wizard will automatically import the key.

   g. Click **Next**. The Import Key panel is displayed.

   h. Enter a fully qualified path in the **Location of Keystore File** field.

      This field is displayed for all format types.

      Optionally, you can click **Browse** to find the file on the file system.

   i. If prompted, enter the **Password** for the keystore file.

      Note: This field is displayed only for PKCS#12 format.

   j. If prompted, enter the key name in the **Enter the name of the key you want to import** field.

      Note: This field is displayed only for PKCS#12 format files that contain multiple key pairs.

k. Enter a string that names the new key in the **New Key Label** field.

   This field is displayed for all format types.

l. Click **Finish** to exit the wizard

4. When your partner requires client basic authentication, you must retain the user name and password strings. After you have created your federation, and are using the management console to add a partner, the Partner wizard will prompt you to enter these values. You do not need to use these strings in any other way.

Adding a partner to a Liberty federation

Procedure

1. Copy the metadata file from the partner to an easily accessible location on your computer For example, /tmp.

   Note: When the partner also uses Tivoli Federated Identity Manager this file was created on the partner computer by using the management console to export the federation properties.
2. Log in to the IBM Integrated Solutions Console Click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Partners**. The Federation Partner Wizard starts and displays the Select Federation panel.
3. Select the radio button next to the Liberty federation and click **Next**. The Metadata Input panel is displayed.
4. Enter the full path of the metadata file (on the local computer) in the Partner's Liberty Metadata File field and click **Next**. For example:

   ```
   /tmp/libertyfed11_metadata_sp.xml
   ```
5. The next configuration task is controlled by whether the imported metadata contains information about a key or certificate

   Typically, the metadata that you imported contains a key that you must import into an existing keystore.
   - When the imported metadata contains information about a key or certificate, the Partner Key panel is displayed. Continue with step 6
   - When the imported metadata *does not* contain information about a key or certificate, the Server Certificate Validation for SOAP panel is displayed. Continue with step 7
6. Enter the requested information for the Partner key.

   Note: This key or certificate is used to sign Liberty messages, and to sign or validate Liberty tokens. This key is not used for securing SOAP communications over HTTPS.
   a. Select a keystore from the keystore table.
   b. Enter the password in the **Keystore Password** field.
   c. Enter a value in the **Enter a label for your partner's key field**. For example:

      ```
      benefits.example.com
      ```
   d. Select or clear the **Require Partner to Sign Liberty Messages** field. Click **Next**.
7. The next configuration step is determined by whether or not the imported metadata contains a SOAP endpoint that is specified to use HTTPS. Choose one of the following actions: .

- When the imported metadata contains a SOAP endpoint that is specified to use HTTPS, you are prompted to specify the keys or certificates to use. Continue with step 8.
- When the SOAP endpoint does not use HTTPS, you do not have to specify keys or certificates. Continue with step 12

Note: In a typical deployment, you will need to specify keys or certificates for use with the SOAP endpoint. Typical practice for optimal security is to secure this endpoint with HTTPS.

8. When the Server Certificate Validation for SOAP panel is displayed, complete the following steps:
   a. Select a keystore from the **Keystore** pull-down menu.

      Tivoli Federated Identity Manager supplies a **DefaultTrustedKeyStore**. If you are using one of the default CA certificates (based on your agreement with your partner), you can select this keystore. Otherwise, you should access the keystore where you placed the certificate you obtained from your partner, for use with SSL communication between SOAP endpoints.

      In a test or prototype environment, you can select **DefaultTrustedKeyStore**
   b. Enter the password in the **Keystore Password** field.

      The default password for **DefaultTrustedKeyStore** is `testonly`.
   c. Click **List Keys**.
   d. Select the radio button for the certificate you want, as indicated by the value in the **Alias** column in the key table.

      In a test or prototype environment, you can select `testwebseal`
   e. Click **Next**. The Client Authentication for SOAP panel is displayed.
9. You are prompted to specify whether the partner requires either *client certificate authentication* or *client basic authentication*. The partner can require only one of these authentication methods. When you select one of the authentication types on the wizard panel, the panel entries for the other authentication type are deactivated.
   - When the partner requires client certificate authentication, continue with 10.
   - When the partner requires client basic authentication, continue with 11.
10. Specify the values for client certificate authentication
    a. Select the **Partner Requires Client Certificate Authentication** check box.
    b. Select a keystore in the **Keystore** menu.

       This is the keystore where you placed the certificate to be used during client certificate authentication.
    c. Enter the password in the **Keystore Password** field.
    d. Click **List Keys**.
    e. Select the radio button for the appropriate key in the key table. Click **Next**.
    f. Continue with 12.
11. Specify the values for client basic authentication
    a. Select the **Partner Requires Client Basic Authentication** field.
    b. Enter the **Username** and **Password** that you obtained from your partner. Click **Next**.
    c. Continue with 12.
12. The next panel to be displayed will depend on your federation role (identity provider or service provider) and your version of Liberty (1.1 or 1.2). In most

cases, you will specify properties for the Liberty token. Select the following instruction that matches your configuration:

- When adding a service provider partner to an identity provider federation, continue with 13
- When adding an identity provider partner to an service provider federation continue with 14

13. Specify the token module configuration data for adding a service provider partner to an identity provider federation. The required data is identical for Liberty v1.1 or Liberty v1.2. The Liberty v1.1 or v1.2 Token Module Configuration panel is displayed.
    a. Specify the types of attributes to include in the Liberty token in the **Include the following types of attribute types (a "*" means include all types)** field.

       You can accept the default entry of asterisk (*) to include all types, or specify attribute types.
    b. Click **Next**. Continue with 12 on page 333.
14. Specify the token module configuration data for adding an identity provider partner to a service provider federation. Select the action that matches the version of Liberty protocol (v1.1 or v1.2).
    - When adding an identity provider partner to an service provider federation, using Liberty v1.1, no token module configuration is required. The Identity Mapping panel is displayed. Continue with 15.
    - When adding an identity provider partner to an service provider federation, using Liberty v1.2, complete the following steps:

    a. You can optionally supply a value for the **Username for anonymous users** field. If you are not using this Liberty feature, you can leave this field blank. .
    b. Click **Next**. The Identity Mapping panel is displayed. Continue with step 15.
15. Choose the following action that matches your use of an identity mapping rule:
    - Leave the identity mapping rule blank when you want to use the default identity mapping rule that you entered in the federation creation wizard. Click **Next**.
    - If you have a customized mapping file for use with this partner, enter the file path to the file. Click **Import**. Click **Next**.

    The Summary panel is displayed.
16. Verify that the settings are correct and click **Finish**.

    The Add Partner Complete panel is displayed.
17. Click **Enable Partner** to activate this partner.

    The partner has been added to the federation, but is disabled by default as a security precaution. You must enable the partner.

Configuring the alias service for Liberty

About this task

The alias service must be configured to operate with the same user registry as the Tivoli Federated Identity Manager management service. These instructions describe configuration of IBM Tivoli Directory Server LDAP.

Procedure

1. "Creating an LDAP suffix for the alias service"
2. "Configuring LDAP server settings"

Creating an LDAP suffix for the alias service

About this task

You must create an LDAP suffix `cn=itfim` to enable the alias service to access the LDAP user registry.

Procedure

1. Stop the IBM LDAP server.

   UNIX
   ```
   # ibmdirctl -D cn=root -w passw0rd stop
   ```
   Windows
   Use the Services icon.
2. Add the suffix:
   ```
   # idscfgsuf -s "cn=itfim"
   ```
3. Start the IBM LDAP server.

   UNIX
   ```
   # ibmdirctl -D cn=root -w passw0rd start
   ```
   Windows
   Use the Services icon.
4. Use **ldapmodify** to update the LDAP schema file. For example, on UNIX or Linux:
   - IBM Tivoli Directory server:
     ```
     ldapmodify -D cn=root -w passw0rd -f
       /opt/IBM/FIM/etc/itfim-secuser.ldif
     ```
   - Sun ONE Directory server:
     ```
     ldapmodify -D cn=root -w passw0rd -f
        /opt/IBM/FIM/etc/itfim-secuser-sunone.ldif
     ```

Configuring LDAP server settings

About this task

The alias service is used by the Liberty protocol. The alias service communicates with the user registry server (LDAP) to manipulate user identity information. You must configure the alias service with the correct LDAP settings.

Procedure

1. Click **Tivoli Federated Identity Manager → Domain Management → Alias Service Settings**. The Alias Service Settings panel is displayed.
2. In the Root Suffix field, under **LDAP Search Settings**, specify the property for the alias service to use when searching the LDAP user registry.

Table 104. LDAP Search property

Property	Description
Root suffix	Specifies the root suffix where alias settings are written. This property can have one value (suffix) only. For example: `cn=itfim`

3. Specify communication properties for the alias service to use when communicating with LDAP servers. Use the menu choices in the **LDAP Environment** portion of the window to specify communication properties.

Table 105. LDAP environment properties

Property	Description
SSL Enabled	Use this check box to specify whether communication between the alias service and the LDAP servers should be secured using Secure Socket Layer (SSL). If the LDAP servers are configured to use SSL, the alias service must use SSL when communicating with them. Select **SSL Enabled** when using SSL. Clear the **SSL Enabled** check box when not using SSL.
Keystore	When the **SSL Enabled** check box is selected, select a keystore from the **Keystore** menu list. The selected keystore is the name of the trusted keystore containing the CA certificate of the LDAP server. Note that the certificate authority certificates for all LDAP servers must be in the same keystore.

4. Specify configuration parameters for each LDAP server. Use the **LDAP Servers** portion of the window to configure properties for LDAP servers used by the alias service.

   You can perform several configuration actions from this section of the window. For each LDAP server, you can specify values for a number of configuration properties.

   - Click **Add** to activate the LDAP configuration fields for the selected server.
   - Click **Save** to save the LDAP properties that you entered into the configuration fields for a server. When you save the properties, the console inserts the host name and port number into the **LDAP hosts** box.

Table 106. LDAP server properties

Property	Description
LDAP Hostname	The **LDAP Hosts** box lists the configured servers in order of preference. The alias service tries first to contact the server at the start (top) of the list. If that contact is unsuccessful, the alias service attempts to contact the next server on the list. Use the up and down arrows located on the right side of the box to move individual LDAP servers higher or lower in the order of priority.

Port	The port on which the LDAP server listens.
	Default port for non-SSL communication:
	`389`
	Default port for SSL communication:
	`636`

Bind DN	The distinguished name (DN) that the alias service uses to bind to the LDAP server. Default value: `cn=root`
Bind Password	The password for the DN specified in the **Bind DN** field.
Key Name	The name of the encryption key to use when establishing SSL communication. Select a key name from the list of names. The names on the list are obtained from the keystore that is specified in the **Keystore** field in the **LDAP environment** portion of this configuration window.

Table 106. LDAP server properties (continued)

Property	Description
Minimum number of connections	The initial number of connections (binds) for the alias service to establish to the LDAP server. The minimum valid number is zero (0). The maximum valid number is limited only by the maximum value supported by the data type. The default value is 2. Use the default value unless you have a specific need to increase it.
Maximum number of connections	The maximum number of connections (binds) for the alias service to establish to the LDAP server. The maximum valid number is limited only by the maximum value supported by the data type. The default value is 10. Use the default value unless you have a specific need to increase it.

5. Click **OK** to save configuration properties and exit from the window.

Chapter 26. Planning a WS-Federation single sign-on federation

You will need to specify values for federation properties when configuring WS-Federation.

WS-Federation protocol defines a standardized, multi-vendor Web-based single sign-on solution based on a collection of integrated Web Services (WS*) standards including WS-Security, WS-Trust, and WS-Federation. When you configure Tivoli Federated Identity Manager, you'll select the WS-Federation Passive Profile.

You should be familiar with the WS-Federation standards documents before implementing a single sign-on federation. The standards specify data exchange and message processing. You should understand what information you are required to provide to your business partners, and what information your partner must provide to you.

Web Services Federation Language (WS-Federation):
`http://wwww.ibm.com/developerworks/library/ws-fed`

The Federation wizard will prompt you to supply values for a number of properties. Most of them can be modified later, after federation creation.

The choice of profile (or profiles) to use is based on both business policy decisions and network security architecture. Federation partners must agree on the profile choices in order to enable user single sign-on across the federation. The choice must be made prior to configuring the federation.

SAML 2 supports a unique range of single sign-on profiles. The profiles extend beyond specifications for achieving federated single sign-on, and can include other functions such as single logout and federation termination.

Identity provider and service provider roles

Each partner in a federation has a role. The role is either **Identity Provider** or **Service Provider**.

- Identity provider

  An identity provider is a federation partner that vouches for the identity of a user. The Identity Provider authenticates the user, and provides an authentication token to the service provider.

  The identity provider either directly authenticates the user, such as by validating a user name and password, or by indirectly authenticating the user, by validating an assertion about the user's identity, as presented by a separate identity provider.

  The identity provider handles the management of user identities in order to free the service provider from this responsibility.
- Service Provider

  A service provider is a federation partner that provides services to end user. Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on identity providers to assert the identity of a user, and rely on identity providers to manage user identities for the federation.

Service providers can maintain a local account for the user, which can be referenced by an identifier for the user.

WS-Federation single sign-on profiles

The single sign-on profiles enables a client using a Web browser to achieve single sign-on access to resources within a WS-Federation 1.0 federation. Typically the user wants to access a resource provided by a service provider, and must authenticate with an identity provider in order to be granted that access. The profile provides a mechanism for the Web user to obtain an authentication assertion that can be used to establish a security context within the federation. Establishment of the security context enables a user to access multiple resources within the federation without having to authenticate more than once.

WS-Federation support two profiles for use with single sign-on sessions:

Browser POST

Browser POST uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

WS-Federation supports browser POST by default. No configuration is required.

Single logout

This profile terminates all login sessions within the federation for a specified user. WS-Federation supports single logout by default. No configuration is required.

WS-Federation single sign-on properties

WS-Federation Realm

The name of the WS-Federation Realm. This name is the unique identifier for this instance of Tivoli Federated Identity Manager The Realm name is included in assertions that are sent to federation partners. Partners rely on finding a known (defined) Realm name in order to accept the assertions. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/wsfed/wsf
```

In the example above, the string `wsfed` is the name of the federation. The endpoint is automatically created. You can accept the default name.

WS-Federation Endpoint

The endpoint for all requests for WS-Federation services. A default value is provided. For example:

```
https://idp.ibm.com/FIM/sps/wsfed/wsf
```

In the example above, the string `wsfed` is the name of the federation. The endpoint is automatically created. You can accept the default name.

WS-Federation token properties

When you create a single sign-on federation, you must configure an instance of a security token module for the federation. The token module corresponds to a security token type that defines the format for the encrypted token that contains user credential information.

The token is exchanged between the identity provider and service provider as part of the authentication and authorization services for the processing of each user access request.

When you use the federation creation wizard to create a WS-Federation single sign-on federation, the SAML 1 token type is automatically selected for you.

When you configure an identity provider, you will be prompted to specify token module properties. When you configure a service provider, you do not have to specify any token module properties.

Number of seconds before the issue date that an assertion is considered valid
Specified during token configuration on identity provider only. Default value 60 seconds. There is no minimum or maximum enforced.

Amount of time the assertion is valid after being issued (seconds)
An integer value that specifies the number of seconds that the assertion remains valid. The default value is 60 seconds. There is no minimum or maximum enforced. Specified during token configuration on identity provider only.

WS-Federation identity mapping

The federation creation wizard will prompt you to specify either an XSLT mapping rule file or a custom mapping module instance.

The XSLT mapping file or custom mapping module instance must be prepared before you configure the federation.

XSLT Transformation for Identity Mapping
Selection of this button in the wizard indicates that you will provide an XSL file containing the identity mapping. Enter the name of a file on the local file system.

Custom Mapping Module Instance
Selection of this button in the wizard indicates that you will provide a custom mapping module instance to use instead of an XSL file. You will be prompted to enter any configuration properties that your custom mapping module instance requires.

Mapping a Tivoli Access Manager credential to a SAML 1 token

This scenario occurs when messages are exchanged between partners in a SAML 1.0, SAML 1.1, or WS-Federation single sign-on federation, and user identity information is managed by Tivoli Access Manager. When a user request is received (for example, for access to a remote resource) the trust service contacts Tivoli Access Manager and obtains a Tivoli Access Manager credential for the user identity.

In this scenario, the trust service Tivoli Access Manager credential module operates in validate mode. In this mode, it converts the Tivoli Access Manager credential to an Input STS universal user document (In-STSUUSER). The In-STSUUSER that is created from the Tivoli Access Manager credential module has all of the information from the credential, as shown in Table 107. This information is available for possible use by the trust service module that will build the outgoing token.

Table 107. In-STSUUSER entries generated from a Tivoli Access Manager credential

Tivoli Access Manager credential	In-STSUUSER element
User ID	Principal Attr: name
Domain	Principal Attr: domain
Registry ID	Principal Attr: registryid
User UUID	Principal Attr: uuid
Group Name	Group Name
Group Registry ID	Group Attr: registryid
Group UUID	Group Attr: uuid
Other credential entries xxx	Attrlist Attr: xxx

The trust service consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

Next, the identity mapping module converts the In-STSUUSER into an Output STS universal user (Out-STSUUSER). The Out-STSUUSER must contain the information needed by the Tivoli Access Manager SAML token module to generate a SAML token.

The Out-STSUUSER must contain the following information in order for the SAML token module to be able to generate a valid SAML token:

Table 108. Out-STSUUSER entries used to generate a SAML token

Out-STSUUSER element	SAML Token Information	Required?
Principal Attr: Name	AuthenticationStatement/Subject/NameIdentifier	Required
Attribute List	Additional custom attributes	Optional

The mapping module is responsible for:

1. Mapping Principal Attr Name in In-STSUUSER to a Principal name entry in the Out-STSUUSER.

   The type must be valid for SAML. For example:

   `urn:oasis:names:tc:SAML:1.0:assertion#emailAddress`

   Figure 45 on page 343 shows a sample mapping rule file from the demonstration application mapping file, ip_saml_10.xsl.

```
<!--
 This template replaces the entire Principal element with one that contains
 just the email address (from the ivcred tagvalue_email) and the data type
 appropriate for SAML.
-->
<xsl:template match="//stsuuser:Principal">
 <stsuuser:Principal>
  <stsuuser:Attribute name="name"
                  type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
   <stsuuser:Value>
    <xsl:value-of
    select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_email']
            [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
   </stsuuser:Value>
  </stsuuser:Attribute>
 </stsuuser:Principal>
</xsl:template>
```

Figure 45. XSL code sample showing mapping of a value from a Tivoli Access Manager credential into a Principal name for a SAML token

2. Setting the authentication method to the "password" mechanism, regardless of the value obtained from the Tivoli Access Manager credential. This action is required by the SAML standard.

   Figure 46 shows a sample mapping rule file from the demonstration application mapping file, ip_saml_10.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
               ....
    <!-- First the authentcation method attribute -->
   <stsuuser:Attribute name="AuthenticationMethod"
                           type="urn:oasis:names:tc:SAML:1.0:assertion">
    <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
   </stsuuser:Attribute>
               ....
               ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 46. XSL code sample showing assignment of authentication method as an Attribute for a SAML token

3. Populating the attribute statement of the SAML assertion with the attributes in the AttributeList in the In-STSUUSER. This information becomes custom information in the SAML token.

   There can be custom attributes that are required by applications that will make use of information that is to be transmitted between federation partners.

   Figure 47 on page 344 shows the mapping of custom attributes in the sample mapping file for the Tivoli Federated Identity Manager demonstration application.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
              ....
       <!-- Now the commonName attribute -->
   <stsuuser:Attribute name="commonName"
                           type="http://example.com/federation/v1/commonName">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_name']
             [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>

   <!-- Now the ssn attribute -->
   <stsuuser:Attribute name="ssn"
                           type="http://example.com/federation/v1/namevalue">
    <stsuuser:Value>
     <xsl:value-of
      select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_ssn']
             [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
               ....
    </stsuuser:AttributeList>
 </xsl:template>
```

Figure 47. XSL code sample showing assignment of optional attributes for a SAML token

4. Note that the GroupList element of the In-STSUUSER is not read by the SAML token module. However, information in this element can optionally be used to populate custom attributes of the Out-STSUUSER.

   Figure 48 shows the optional assignment of a GroupList value to an attribute. This code sample is from the demonstration application mapping file ip_saml_10.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
       <!-- Now the role attribute (can be multi-valued) -->
   <stsuuser:Attribute name="role" type="http://example.com/federation/v1/role">
    <xsl:for-each select="//stsuuser:GroupList/stsuuser:Group">
     <stsuuser:Value>
      <xsl:value-of select="@name" />
     </stsuuser:Value>
    </xsl:for-each>
   </stsuuser:Attribute>
              ....
    </stsuuser:AttributeList>
 </xsl:template>
```

Figure 48. XSL code sample showing optional assignment of GroupList value to an attribute for a SAML token

Mapping a SAML 1 token to a Tivoli Access Manager credential

The service provider receives a SAML token. The SAML token module, operating in validate mode, creates an In-STSUUSER document from the SAML token. Table 109 on page 345 shows the information from the token that is converted into an In-STSUUSER document.

Note: This topic applies to both SAML 1.0 and SAML 1.1 tokens.

Table 109. SAML token information that is converted into a STS universal user document

SAML Token Information	In-STSUUSER element	Required for Out-STSUUSER?
AuthenticationStatement/Subject/ NameIdentifier	Principal Attr: Name	Required
Additional custom attributes	Attribute List	Optional

Note that the SAML token module does not populate the GroupList element in the In-STSUUSER document.

The trust service must convert this information to a Tivoli Access Manager credential, in order to make an authorization decision on the request from the user identity. The identity mapping module converts the In-STSUUSER data into an Out-STSUUSER XML file.

- The NameIdentifier is used to populate the `name` attribute of the Principal.

  Figure 49 shows the assignment of a set value for the Principal name. This code sample is from the demonstration application mapping file sp_saml_1x.xsl

```
<!--
  This will replace the principal name (which was the email address in
  the SAML assertion) with the user "me_chris".
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
           <stsuuser:Value>me_chris</stsuuser:Value>
           </stsuuser:Attribute>
 </xsl:template>
```

Figure 49. XSL code sample showing assignment of a value for the Principal name for a SAML token.

- Other information from the token is used to populate Attributes in the Attribute List.

  Figure 50 on page 346 shows the optional assignment of additional values to attributes. This code sample is from the demonstration application mapping file sp_saml_1x.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
         ....
      <!-- The tagvalue_name attribute -->
   <stsuuser:Attribute name="tagvalue_name"
                           type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuuser:Value>
    <xsl:value-of
  select="//stsuuser:AttributeList/stsuuser:Attribute[@name='commonName']
      [@type='http://example.com/federation/v1/commonName']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>

   <!-- The tagvalue_ssn attribute -->
   <stsuuser:Attribute name="tagvalue_ssn"
                           type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuuser:Value>
     <xsl:value-of
              select="//stsuuser:AttributeList/stsuuser:Attribute[@name='ssn']
         [@type='http://example.com/federation/v1/namevalue']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 50. XSL code sample showing optional assignment of attributes for a SAML token.

Chapter 27. Configuring a WS-Federation single sign-on federation

To configure a WS-Federation single sign-on federations, you must create the federation, add your partner to your federation, and provide your partner with configuration information from your new federation.

About this task

Complete the instructions in each of the following sections:

Procedure

1. "Creating a WS-Federation single sign-on federation"
2. "Configuring WebSEAL as the point of contact server" on page 348
3. "Exporting WS-Federation properties" on page 349
4. "Obtaining configuration information from a WS-Federation partner" on page 350
5. "Adding a partner to your WS-Federation single sign-on federation" on page 352

Creating a WS-Federation single sign-on federation

About this task

- Chapter 26, "Planning a WS-Federation single sign-on federation," on page 339.

Complete the following steps:

Procedure

1. Log in to the Integrated Solutions Console and click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets are displayed. The Federations portlet displays several action buttons.
2. Click **Create**. The Federation Wizard starts. The General Information panel is displayed.
3. Enter a name for the federation and select a role. Click **Next**.
4. Enter the contact information and click **Next**.
5. Select the WS-Federation Passive Protocol and click **Next**. The Point of Contact Server panel is displayed.
6. Enter the point of contact address and click **Next**.
7. Choose one:
   - When configuring a service provider, the next step is identity mapping. Continue with step 8
   - When configuring an identity provider, the Configure Security Token panel is displayed. Specify the requested token properties and click **Next**.

     See Chapter 26, "Planning a WS-Federation single sign-on federation," on page 339.
8. The Identity Mapping Options panel is displayed. Select one of the radio buttons.

- Use XSL Transformation for Identity Mapping

  Indicates that you will provide an XSL file containing the required identity mapping.

  a. When you select this choice and click **Next**, the Identity Mapping panel is displayed. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

     This is a file that you have prepared prior to this installation.

     Optionally you can click the **Browse** button to locate the file on the local file system.

  b. Click **Next**.

     An error is displayed if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).

- Use Custom Mapping Module Instance

  Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

  a. When you select Use Custom Mapping Module Instance, a table of Module Instances is displayed. Select the radio button for the module instance to use and click **Next**.

  b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel displays a message indicating that there are no properties to configure for the specified module instance.

9. The Summary panel is displayed. Verify that the configuration settings are correct and click **Finish**. The Create Federation Complete portlet is displayed.
10. Click **Restart WebSphere**.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See:

- "Configuring WebSEAL as the point of contact server"

Configuring WebSEAL as the point of contact server

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the WS-Federation single sign-on federation.

Before you begin

The federation wizard provides a button that you can use to obtain a configuration utility.

About this task

You must obtain the utility and run it. Complete the following steps:

Procedure

1. From the management console, click **Download Tivoli Access Manager Configuration Tool**
2. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.

3. Return to the management console, and Click **Done** to return to the Federations panel.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation we will complete other tasks first.
4. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -cfgfile webseald-instance_name.conf
    -action tamconfig
   ```

   You will need to know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -cfgfile webseald-default -action tamconfig
```

For more information, see:
- Appendix A, “tfimcfg reference,” on page 491

What to do next

The next task is to manually export your WS-Federation properties to your partner. See “Exporting WS-Federation properties.”

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log in to the administration console.
2. Click Tivoli Federated Identity Manager > Manage Configuration > Point of Contact
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Exporting WS-Federation properties

About this task

When you want to join a federation hosted by another business partner, you must supply your federation configuration properties. For WS-Federation, you must manually prepare a file that contains the configuration properties. Give this file to your federation partner.

Procedure

1. Log in to the management console. Click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**.
2. The Federations panel is displayed. Select your WS-Federation single sign-on federation from the table.
3. Display the federation properties. Obtain the properties listed in "WS-Federation properties to exchange with your partner"
4. Deliver the file to your partner, in the manner specified in the business agreement between your company and your partner's company.

What to do next

You will need to provide this file to your partner, when your partner wants to add your configuration information to their WS-Federation single sign-on federation.

Obtaining configuration information from a WS-Federation partner

You must obtain configuration information from your WS-Federation your partner.

Before you begin

When you want to add your business partner as a partner to your WS-Federation single sign-on federation, you must obtain from them the necessary configuration information about their WS-Federation single sign-on federation.

Your partner must have already installed and configured a WS-Federation single sign-on federation. The partner's federation plays the opposite role to your federation. For example, when your federation is configured as an identity provider, your partner's federation is configured as a service provider.

You obtain the information by having your partner manually assemble the configuration properties for their federation. The partner must then provide the information to you through a method that have been agreed upon as part of the business agreement between you and your partner.

About this task

Complete the following task:

Procedure

1. Your partner must use the management console to collect the properties for their federation. The partner should provide you with the properties listed in "WS-Federation properties to exchange with your partner."
2. You partner should deliver the file to you, in the manner specified in the business agreement between your company and your partner's company.

WS-Federation properties to exchange with your partner

Federation properties

Table 110. WS-Federation properties

Property	Description
Federation name	A character string that names the federation

Table 110. WS-Federation properties (continued)

Property	Description
Role	Identity provider or service provider
Protocol	WS-Federation Passive Profile
Contact Information	
Company name	The name of the company that created this federation. Required.
Company URL	A URL for the company that created this federation. Optional.
First Name and Last Name	The name of the person who serves as contact for other companies in the federation. Optional.
Email address	The e-mail address of the person who serves as contact for other companies in the federation. Optional.
Phone number	The telephone number of the person who serves as contact person for other companies in the federation. Optional.
Contact Type	A string that describes a business role type such as technical or support. Optional

WS-Federation data

Table 111. WS-Federation data

Property	Description
WS-Federation Realm	The unique name of the WS-Federation Realm. For example: `https://idp.example.com/FIM/sps/wsfed/wsf`
WS-Federation Endpoint	The partner's endpoint for all requests for WS-Federation services. For example: `https://idp.example.com/FIM/sps/wsfed/wsf`
Maximum Request Lifetime (in seconds)	The maximum length of time, in seconds, that a request or message that is received from a WS-Federation partner is valid.

SAML token module configuration

Table 112. SAML token module properties

Header	Header
Enable the Signing of Assertions	Indicates whether the identity provider will sign assertions before sending them to the service provider partner.
Select Key for Signing Assertions	The name of the key to use when signing assertions. Specified for a service provider partner.
Include the following attribute types (a '*' means include all types)	The types of attributes to include in the SAML token module. Specified for a service provider partner.

Table 112. SAML token module properties (continued)

Header	Header
Enable signature validation	When selected, indicates that the service provider will validate the signature on assertions received from the identity provider partner. Specified for an identity provider partner
Select Validation Key	The name of the key to use when validating signatures. Specified for a identity provider partner.

Adding a partner to your WS-Federation single sign-on federation

You can use the administration console to add a partner to a WS-Federation single sign-on federation.

About this task

The configuration steps are the same for adding all partners. The configuration properties differ for identity provider and service provider partners. The Partner Wizard prompts you for the necessary properties.

Procedure

1. Log in to the IBM Integrated Solutions Console. Click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Partners**.
2. The Federation Partners panel is displayed. Click **Create**. The Select Federation panel is displayed.
3. Select the federation to which you would like to add a partner. Click **Next**. The Contact Information panel is displayed.
4. Enter the Contact properties and click **Next**.

   The company name is required. The other fields are optional. For more information, see . The WS-Federation Data panel is displayed.
5. Enter the requested properties and click **Next**.
6. The Configure Security Token panel is displayed. Enter the configuration properties for the federated security token.

   The configuration properties are specific to the partner role:

   - When adding an identity provider partner:
     a. When assertions should be signed click **Enable the Signing of Assertions**. When you select this check box, you must specify a key for signing assertions. Select the **Keystore**, enter the **Keystore Password**, click **List Keys** and select the key from the key table.
     b. Optionally specify attributes in the field: **Include the following attribute types (a '*' means include all types**
     c. Click **Next**.
   - When adding a service provider partner:
     a. When signatures should be validates click **Enable Signature Validation**. When you select this check box, you must specify a key to use for validating signatures. Select the **Keystore**, enter the **Keystore Password**, click **List Keys** and select the key from the key table.
     b. Click **Next**.

7. The Identity Mapping Options panel is displayed. Select one of the radio buttons.
   - Use XSL Transformation for Identity Mapping

     Indicates that you will use an XSL file to provide any required identity mapping.

     a. When you select this choice and click **Next**, the Identity Mapping panel is displayed. Leave the identity mapping blank when you want to use the default identity mapping rule that you entered in the federation creation wizard. When you want to override the default mapping rule with a rule specific to this partner, enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

        For more information on mapping rules files, see.

        Optionally you can click the **Browse** button to locate the file on the local file system.

     b. Click **Next**.
   - Use Custom Mapping Module Instance

     Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

     a. When you select Use Custom Mapping Module Instance, a table of Module Instances is displayed. Select the radio button for the module instance to use and click **Next**.
     b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel displays a message indicating that there are no properties to configure for the specified module instance.

   The Summary panel is displayed.
8. Verify that the settings are correct and click **Finish**. The Add Partner Complete panel is displayed.
9. Click **Enable Partner** to activate this partner.

   The partner has been added to the federation, but is disabled by default as a security precaution. You must enable the partner.

Part 3. Web services security management configuration

The topics in the Configuration section provide a step-by-step guide to configuring Web services security management for Tivoli Federated Identity Manager.

Read the overview section first:

Chapter 28, “Web services security management configuration,” on page 357

Chapter 28. Web services security management configuration

Configuration of Web services security management starts with the establishment of a Tivoli Federated Identity Manager domain. When the domain is established, you can configure the Web services security management component.

Configuration of Web services security management consists of these steps:

1. Configuration of a Tivoli Federated Identity Manager domain.

   The deployment of a Tivoli Federated Identity Manager scenario requires the creation of a Tivoli Federated Identity Manager domain.

   You must create and configure a domain before you can configure the Web services security management component.

   See Chapter 1, "Domain configuration," on page 3.
2. Configuration of the Web services security management component.

   The component can be configured in many different ways, to reflect the deployment scenarios. Configuration is described in detail in the *IBM Tivoli Federated Identity Manager Web Services Security Management Guide*.

Part 4. Configuring security token service

The topics in the Configuration section provide a step-by-step guide to configuring a security token service as part of an integrated solution for management of user identities in a distributed network environment.

This section describes the deployment of a Kerberos delegation security token service module as support for a Kerberos junction solution provided by combining Tivoli Federated Identity Manager with Tivoli Access Manager for e-Business WebSEAL, along with WebSphere and additional products.

First read the overview of the deployment scenario:

Chapter 29, “Kerberos constrained delegation overview,” on page 361

Chapter 29. Kerberos constrained delegation overview

Tivoli Federated Identity Manager provides a security token service (STS) that can exchange security token formats. This function is used to move user credential information between different token formats, as needed for different applications.

The STS is an integral part of the Tivoli Federated Identity Manager single sign-on solutions, but can also be used standalone. This ability to be used standalone allows Tivoli Federated Identity Manager to be integrated into a variety of heterogeneous network deployments.

One such deployment is an environment that uses Microsoft Windows integrated authentication (SPNEGO) in conjunction with Kerberos tickets. In this environment, Tivoli Federated Identity Manager can be deployed to take user credentials and convert them to the necessary Kerberos format.

To enable this capability, Tivoli Federated Identity Manager includes a security token service module specifically for Kerberos constrained delegation. The Kerberos delegation module facilitates the issuing of Kerberos Constrained Delegation application service tickets, also known as Service for User To Proxy (S4U2Proxy).

The module supports only the *issuing* of tokens, and *only* Windows Kerberos application service tickets through the Constrained Delegation model.

A primary feature of the Kerberos constrained delegation model is that the password of the end-user for whom the Kerberos service ticket is to be obtained need not be known by the application generating the ticket. In this case the application is WebSphere plus Tivoli Federated Identity Manager. The application needs to know only the username of the end user, and the service principal name (SPN) of the destination Kerberos service.

The Kerberos constrained delegation STS module is primarily intended to enable Tivoli Access Manager WebSEAL to support single sign-on across Kerberos junctions. These junctions are junctions to a Web server that is configured for Integrated Windows Authentication (SPNEGO). WebSEAL can maintain a user session using whatever authentication mechanism it chooses, and then connect to a Web server (for example, IIS) by using the SPNEGO authentication flow. This authentication flow uses a Kerberos ticket.

The use of Kerberos credentials for single signon to junctions provides the following capabilities:

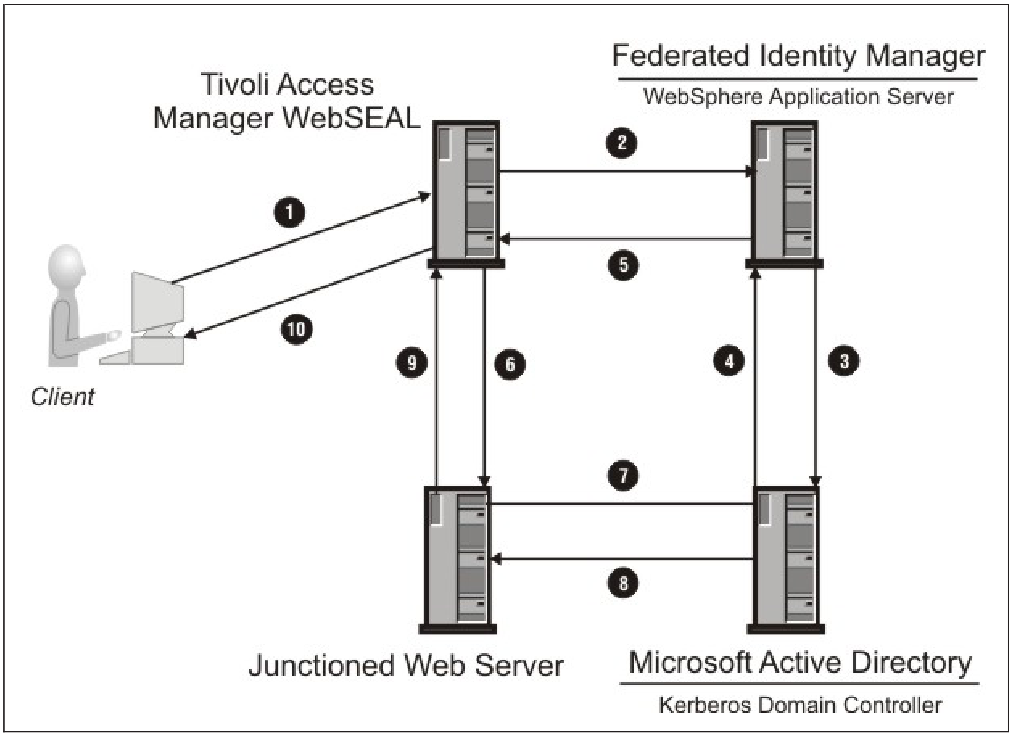
- Kerberos credentials are easily utilized by ASP.NET Web applications without requiring special code to be deployed.
- Kerberos credentials can be forwarded across applications while maintaining a cryptographic signature, providing stronger security.

Note: This module is different from the Tivoli Federated Identity Manager native Java Kerberos STS module. The Java Kerberos module supports the generic issuing and validation of other Kerberos tickets.

More information on the Windows Kerberos extensions can be found at:

- http://technet2.microsoft.com/WindowsServer/en/Library/c312ba01-318f-46ca-990e-a597f3c294eb1033.mspx
- http://msdn2.microsoft.com/en-us/library/aa480585.aspx
- http://msdn.microsoft.com/msdnmag/issues/03/04/SecurityBriefs/

Overview of Kerberos constrained delegation with WebSEAL junctions


Federated Identity Manager
WebSphere Application Server
Tivoli Access
Manager WebSEAL
2
1
5
10
Client
9
6
4
3
7
8
Junctioned Web Server
Microsoft Active Directory
Kerberos Domain Controller

Tivoli Federated Identity Manager uses the Kerberos constrained delegation security token service (STS) module to generate Kerberos tokens. WebSEAL retrieves the Kerberos tokens by delegating the token request to the STS module.

The diagram above shows a sample deployment of the applications involved in achieving this type of single sign-on. The diagram also shows how the messages flow between the different physical components.

1. Client uses the standard Tivoli Access Manager authentication process to authenticate to WebSEAL over HTTPS or HTTP and requests an object on the junctioned server. WebSEAL authorizes the request from the client, and determines that a Kerberos ticket is needed to access the junctioned application.
2. WebSEAL generates a WS-Trust issue key message intended for the Tivoli Federated Identity Manager server. A single WS-Trust issue key can be used to request multiple Kerberos tokens. WebSEAL opens a connection to the WebSphere server running Tivoli Federated Identity Manager. WebSEAL sends the SOAP request to the WebSphere server.
3. The Tivoli Federated Identity Manager server running on the WebSphere server verifies that the WebSEAL server is authorized to invoke the security token service (STS). The STS then invokes a Tivoli Federated Identity Manager trust module to request the configured number of Kerberos tickets for the

junctioned Web server on behalf of the client. The trust module communicates with the Active Directory domain controller using Kerberos, over TCP or UDP port 88.

4. The Active Directory domain controller verifies that the Tivoli Federated Identity Manager server is authorized to request tickets for the junctioned Web server on behalf of the user. The Active Directory domain controller returns the configured number of Kerberos tickets to the Tivoli Federated Identity Manager runtime.
5. The Tivoli Federated Identity Manager runtime returns the tickets as a SOAP response to the WebSEAL server.
6. The WebSEAL server caches the Kerberos tickets and forwards one of the tickets along with the client request to the junctioned Web server over either HTTP or HTTPS.
7. The junctioned Web server requests validation of the Kerberos ticket from the Kerberos domain controller (KDC). The KDC is shown here as being the same system as the Active Directory server.
8. The KDC verifies that the Kerberos ticket is valid. The Kerberos ticket is used as proof of the client identity, which may also be used for further authorization checks.
9. The junctioned Web server returns an HTTP response to WebSEAL.
10. WebSEAL returns the HTTP response to the client.

On each subsequent request from the same client on the same login session to the same junctioned Web server, a new Kerberos ticket is sent along with the request to the junctioned Web server. The new Kerberos ticket is either taken from the WebSEAL cache of Kerberos tickets or a request is sent to the WebSphere server running Tivoli Federated Identity Manager to obtain a new set of Kerberos tickets.

Deployment overview

Software prerequisites

- Tivoli Federated Identity Manager must run on Windows 2003 Server Service Pack 2 or later.

  The service pack is required due to a known memory leak in the Windows Isass.exe process on earlier versions. See http://support.microsoft.com/kb/907524/
- The WebSEAL server can run on any supported platform.

  The WebSEAL server does *not* need to be part of the Active Directory domain.
- WebSphere can be deployed either in standalone mode or in cluster mode. All WebSphere servers in the cluster should be installed on Windows systems, and should be part of the domain.
- When the Tivoli Access Manager users are stored in Active Directory, the Tivoli Access Manager policy server must be on Windows and be a member of the domain.
- All domain controllers in the Active Directory domain should run at the Windows Server 2003 functional level.
- The Tivoli Federated Identity Manager support for Kerberos delegation modules is not included in the Tivoli Federated Identity Manager Business Gateway product.

Deployment task overview

1. Enable integrated Windows authentication
2. Configure Active Directory and WebSphere for constrained delegation
3. Install and configure a Tivoli Federated Identity Manager domain and runtime
4. Configure a Kerberos module instance and trust chain for the Kerberos constrained delegation STS module
5. Configure WebSEAL to support a Kerberos junction

Table 113. Example server hostnames used in this documentation

Server role	Example Value
Backend server (junctioned Web server)	mydataserver.example.com
WebSEAL server	websealhost.example.com
Active Directory hostname	activedirectoryhost.example.com

Chapter 30. Enabling integrated Windows authentication

These instructions describe how to configure Microsoft IIS for SPNEGO authentication.

Before you begin

These instructions assume that you have Windows Server 2003 deployed with Active Directory. These steps must be completed before you can set up constrained delegation.

Procedure

1. On the domain controller, select **Start -> Programs > Administrative Tools -> Active Directory Users and Computers**
2. Create a user that acts as a proxy for the IIS server. For example, `iisuser`. Specify the user password as `never expires`.
3. Open a command prompt.
   a. Change directory to `C:\Program Files\Support Tools`.
   b. Enter the appropriate **ktpass** command.

      Syntax for ktpass:

      ```
      ktpass -princ HTTP/IIS_server_name.domain_name@DOMAIN_NAME
        -mapuser IIS_user_name -mapOp set
      ```

      where:

      - `-princ` is the Principal Name, in the form *`user@REALM`*
      - `-mapuser` maps the -princ value to this use account. This is not done by default.
      - `-mapOp` specifies how to set the mapping attribute: `set` *`set_value`*
4. View the account properties for `iisuser`. Verify that the field **User logon name** is set to the following value:

   ```
   HTTP/IIS_server_name.domain_name
   ```

   For example:

   ```
   HTTP/mydataserver.example.com
   ```
5. Configuring the Application Pool Identity.
   a. On the IIS server system, select **Start -> Programs > Administrative Tools -> Internet Information Service (IIS) Manager**.
   b. Select ***your_server_name/IIS name* -> Programs > Application Pools -> Default App Pool**
   c. Right-click and select **Properties**. Select the identity tab, and specify the domain identity for your IIS user (for example `iisuser`).

   For detailed instructions on the Windows task *Configuring Application Pool Identity with IIS 6.0*, see

   http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/f05a7c2b-36b0-4b6e-ac7c-662700081f25.mspx?mfr=true
6. Open Windows Explorer.
   a. Go to `C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET Files`
   b. Select **Properties**.

c. Select the **Security** tab.

d. Grant domain user `iisuser` full control over the directory

7. Go to the IIS system, and select **Start -> Programs > Administrative tools -> Computer Management**.

   a. Open **Local Users and groups**

   b. Open **groups**

   c. Right click on the local group `IIS_WPG`.

   d. Select properties.

   e. Select Add.

   f. Add the domain user (in our case, `iisuser`) to this local group.

8. On IIS system, Open the server's local security policy. Click **Start -> Run** and enter `secpol.msc`

   a. Expand local polices and browse to User Rights assignment.

   b. Open up the **Log on as Service** right.

      Note that any account or group in this list can logon as a service.

   c. Click **Add User or Group**.

   d. Enter (or browse for) the domain user `iisuser` account.

   e. When the right is granted, reboot the server.

      The system reboot is required because security settings are applied during the startup phase of any Windows 2003 Server machine.

9. On the IIS system, select **Start -> Programs > Administrative Tools -> IIS Manager**.

   a. Open the local computer.

   b. Right-click on the DefaultAppPool.

   c. Select **Recycle** to restart the pool.

10. Open a browser and access `http://`*`web_server`*.

    When this is a new IIS server without existing content, you should see the IIS `Under Construction` page. When the IIS server has content, you should be able to see the content.

11. On the IIS system, select **Start -> Programs > Administrative Tools -> IIS Manager**.

    a. Right-click on Default Web Site

    b. Select Properties and select the Directory Security tab.

    c. Click the **Edit** button next to `Enable anonymous access`, and edit the authentication messages for this resource.

    d. Disable anonymous access.

    e. Enable integrated windows authentication.

    f. Click **OK** and click **OK** again.

12. Open your browser and access http://*web_server*. You are prompted to log in.

13. Enter a valid domain user. For example, user@mydomain.com. When the log in is successful you can view the IIS content.

Chapter 31. Configuring Active Directory and WebSphere for constrained delegation

About this task

The WebSphere node agent that hosts the Tivoli Federated Identity Manager runtime needs to run under a special account in Active Directory in order to have permission to obtain Kerberos tickets for other users and a constrained set of targets. You need to create the account, set the appropriate options, and modify the WebSphere service to use the account before your Kerberos delegation trust chain can work. The following instructions describe how to complete these tasks.

Procedure

1. Verify that DNS is configured correctly on the Active Directory domain controller.

   The DNS server must be configured for both forward and reverse lookups. Each host in the Active Directory domain must be configured to use the Domain Controller's DNS server.

   To verify, use **nslookup** commands for both hostname and IP address on computers in the domain. The results of the **nslookup** commands should show that the domain part of the resolved name is the domain of the domain controller.
2. Ensure that Time Services are running on all machines in the Active Directory domain and that the clocks of all machines are synchronized.
3. Verify that the Windows Server 2003 system (or multiple systems, when deployed in a WebSphere cluster) is configured into an Active Directory domain. The server can optionally be a domain controller.
4. Verify that all domain controllers in the domain are running at the Windows Server 2003 functional level. To do this:
   a. Open the Active Directory Users and Computers control panel.
   b. Right-click on the domain and select **Raise Domain Function Level**.
   c. Select `Windows Server 2003` and click **OK**

   The Raise Domain Functional Level window is displayed. It should contain the messages:

   ```
   Current domain functional level
   Windows Server 2003

   This domain is operating at the highest possible functional level.
   ```
5. On the domain controller, create a user in Active Directory for delegation. The WebSphere server that hosts the Tivoli Federated Identity Manager runtime runs as this user identity.
   a. Create a user. For example, `tfimdeleguser` You can use a different user identity. This user name will be used in these instructions.
   b. Select the **Password never expires** check box.

      Note: You can optionally set the password to expire. If you do, then when you change it in the future, you will also need to reset the password for the WebSphere node agent Windows service.

6. On the domain controller, add the `tfimdeleguser` user to the Domain administrative group. To verify:
   a. Select **Active Directory Users and Computer**
   b. For the domain, click Users and click Domain Admins
   c. Select the Members tab. Verify that the tfimdeleguser is listed as a group member.
7. Ensure that the Microsoft Support Tools are installed on the domain controller. For example to obtain the Windows Server 2003 Service Pack 1 32-bit Support Tools:

   http://www.microsoft.com/downloads/details.aspx?FamilyId=6EC50B78-8BE1-4E81-B3BE-4E7AC4F0912D&displaylang=en
8. On the domain controller, create an service principal name (SPN) for the user `tfimdeleguser`. To do this:
   a. Open a command prompt on the domain controller where the support tools are installed.
   b. Enter the **setspn** command

      The syntax for the command is:

      ```
      setspn -A tfim/<tfim_delegation_user> <tfim_delegation_user>
      ```

      For example:

      ```
       setspn -A tfim/tfimdeleguser tfimdeleguser
      ```
9. On the domain controller, go to **Active Directory Users and Computers** and open the properties for the user `tfimdeleguser`.
   a. Select the Delegation tab.

      Note: If you do not see the Delegation tab, return to the previous step and ensure that the setspn command succeeds
   b. Select the **Trust this user for delegation to specified services only** radio button.
   c. Select the **Use any authentication protocol** radio button.
   d. Click the **Add** button on the Delegation tab. Next, add the target services to which tfimdeleguser can delegate. These are the target services for constrained delegation. In this example, this is the user that the IIS Web server is running as.
   e. Click the **Users or Computers** button to search for particular services.
   f. Select the domain user (service) that the IIS server for the WebSEAL Kerberos junction runs as.

   When you are finished, the Delegation tab should show a target service in the window **Services to which this account can present delegated credentials**.

   For example, the window could show a **Service Type** of `HTTP`, with **User or Computer** showing a host and domain name such as `mydataserver.example.com`

   Select the `HTTP/mydataserver.example.com` entry. Press **OK** to continue.
10. Add the tfimdeleguser to the Windows Authorization Access Groups object. To do this:
    a. Open the **Active Directory Users and Computers** panel.
    b. Select the **Builtin** object under the domain.
    c. Locate the **Windows Authorization Access Groups** object.
    d. Right click and select **Properties**. Select the **Members** tab.

   e. Click **Add** and add the delegation user (in our example, `tfimdeleguser`) as a member.

11. Grant the delegation user (tfimdeleguser) the **Act as part of the operating system** privilege.

    The actual process that must run as a Windows service depends on the WebSphere environment:

    - The service name in a *standalone* environment is the WebSphere Application Server running the Tivoli Federated Identity Manager runtime
    - The service name in a *cluster* environment is the WebSphere Application server running the WebSphere node agent for the Tivoli Federated Identity Manager runtime.

    Note: For a cluster environment, this step must be repeated on all machines hosting a node member of WebSphere cluster running the Tivoli Federated Identity Manager runtime.
    To do this:

    a. Access the menu appropriate for your deployment:
       - On the domain controller, select **Start > Programs > Administrative Tools > Domain Security Policy**.
       - On a non-domain controller computer, select **Start > Programs > Administrative Tools > Local Security Policy**

    b. Expand Local Policies.

    c. Select **User Rights Assignment -> Act as part of the operating system.**

    d. Right-click and select **Properties**.

    e. Click the **Define these policy settings** checkbox

    f. Click **Add user or group** to add the delegation user (tfimdeleguser) to the list of users authorized to act as part of the operating system.

    g. Click **OK**.

12. Grant the delegation user (tfimdeleguser) the necessary privileges:

    - When the Tivoli Federated Identity Manager application is running on a member of the domain, grant the user the permission **Log on as a service privilege** on the local machine.
    - When the Tivoli Federated Identity Manager application is running on the domain controller, grant the user the permission **Log on as a service privilege** on the domain controller

    a. Return to the Security Policy menu opened in the previous step.

    b. Select **User Rights Assignment > Log on as service.**

    c. Right-click and select **Properties**.

    d. Click the **Define these policy settings** checkbox

    e. Click **Add user or group** to add the delegation user (tfimdeleguser) to the list of users authorized to act as part of the operating system.

    f. Click **OK**.

13. Enable the WebSphere process that runs the Tivoli Federated Identity Manager application to run as a Windows service.

    Use the **wasservice** command. Default location:

    ```
    C:\Program Files\IBM\WebSphere\AppServer\bin
    ```

    Example command:

    ```
    C:\Program Files\IBM\WebSphere\AppServer\bin>wasservice -add ndagentwinser
    -servername nodeagent
    -profilePath "C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01"
    ```

```
-wasHome "C:\Program Files\IBM\WebSphere\AppServer"
-logfile "c:\Program Files\IBM\WebSphere\AppServer\profiles\
   Custom01\logs\ws_startserver.log"
-logRoot "c:\Program Files\IBM\WebSphere\AppServer\profiles\
   Custom01\logs\nodeagent"
-restart true
```

Example output from the command:

```
Adding Service: ndagentwinser
  Config Root:
  C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01\config
  Server Name: nodeagent
  Profile Path: C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01
  Was Home: C:\Program Files\IBM\WebSphere\AppServer\
  Start Args:
  Restart: 1
IBM WebSphere Application Server V6.1
   - ndagentwinser service successfully added
```

To obtain a usage message for the **wasservice** command, enter:

```
> WASService.exe
```

without any arguments.

14. If running in a cluster environment, modify the WebSphere service from the previous step to start as the delegation user (tfimdeleguser)
    a. Open the **Services** control panel and locate either the service for the Tivoli Federated Identity Manager runtime or the Tivoli Federated Identity Manager runtime node agent for a cluster environment.
    b. Select the LogOn tab.
    c. Specify the delegation user `tfimdeleguser`
    d. Specify the password for the delegation user. Click **OK**.
15. Restart the WebSphere nodeagent

    This step is required to ensure that the Websphere node manager start the managed nodes under the new identity.
    a. Log on to the WebSphere console.
    b. Click on **Servers -> Application servers** for a standalone environment or **Servers -> Clusters** for a cluster environment
    c. Select the checkbox for the server or cluster to be restarted and press the **Stop** button for a standalone environment or the **Ripplestart** button for a cluster environment
    d. In a standalone environment, after the server has been stopped, select the checkbox for the server or cluster to be restarted and press the **Start** button.

What to do next

Further information:

- Microsoft configuration principles:

  http://technet2.microsoft.com/windowsserver/en/library/c312ba01-318f-46ca-990e-a597f3c294eb1033.mspx?mfr=true
- Configuration instructions:

  http://technet2.microsoft.com/windowsserver/en/library/e5d4cdbd-f071-4a1a-b24e-92713f7fafc11033.mspx?mfr=true
- IBM instructions for configuring WebSphere to run as an account other than **Local System**

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/tsec_actwindows.html

Chapter 32. Tivoli Federated Identity Manager configuration for a Kerberos junction scenario

Before you being configuring Kerberos delegation, ensure that you have created a domain, as described in Chapter 1, “Domain configuration,” on page 3.

Configuration steps:
1. “Planning configuration of the trust chain”
2. Completing the “Worksheet for trust chain configuration” on page 376
3. “Creating a Kerberos constrained delegation module instance” on page 378
4. “Creating a trust chain for Kerberos constrained delegation” on page 379

Planning configuration of the trust chain

To deploy a trust chain for Kerberos constrained delegation, you must complete two tasks:
1. Create an instance of a Kerberos constrained delegation trust service module.
2. Create a trust chain for Kerberos constrained delegation.

Tivoli Federated Identity Manager provides configuration wizards for each task. The wizards prompt you to supply values for the required configuration properties.

Kerberos delegation module instance

The default set of Tivoli Federated Identity Manager trust modules does not include an instance of the Kerberos constrained delegation module type. You must create the instance.

Although it is possible for you to create more than one instance, you should create only one instance for each Tivoli Federated Identity Manager domain. This instance can be used in any module chain that is required.

The reason for the restriction to only one instance is that Kerberos constrained delegation module loads a native DLL (Windows dynamically loaded library) that is shared by all instances of the module. All instances therefore share the same configuration parameters.

When more than one module instance is created, only the *last* module to be initialized determines the size of the user cache created in the native code. To prevent confusion, the best practice is to create only one module instance.

Module Type
This required property is requested on the Module Type panel. The module type to use is:

```
com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule
```

Module Instance name
This required property is requested on the Module Instance Name panel. Supply a string of your choosing. For example:

```
MyKerberosDelegationInstance
```

Module Instance Description

This optional property is requested on the Module Instance Name panel. You can enter a string that describes the instance.

Maximum size of the user credential cache

This required property is requested on the Kerberos Delegation Module Configuration panel. This number determines the number of impersonation handles and user credentials cached in the DLL loaded by the module. The caching is done to improve performance. Set this number to the approximate number of expected concurrent end users of the service for high-volume transactions.

The default setting is 100.

Note: The higher the number, the more memory that will be consumed by Tivoli Federated Identity Manager runtime application.

Kerberos delegation trust chain

Chain Mapping Name

This required property is requested on the Chain Mapping Identification panel. You can specify any name for the chain. For example:

```
ivcred_to_kerberos
```

Chain Description

This optional property is requested on the Chain Mapping Identification panel. The description can be any string.

Create a Dynamic Chain

This property is requested on the Chain Mapping Identification panel. This option is not used with Kerberos delegation trust chains. Deselect this option

Request Type

This required property is requested on the Chain Mapping Lookup panel. Select **Issue Oasis URI**

Lookup Type

Select the radio button **Use Traditional WS-Trust Elements (AppliesTo, Issuer, and Token Type)**.

(AppliesTo) Address

This required property is requested on the Chain Mapping Lookup panel. Enter an **Address** that corresponds to the **applies-to** property in the [tfimsso:*jct_name*] stanza in the WebSEAL configuration file. For example:

```
http://websealhost.example.com/kerbjct
```

(AppliesTo) Service Name

This required property is requested on the Chain Mapping Lookup panel.

This property has two fields.

For the first field, either set this value to asterisk (*) to match all service names, or set it to value of service-name property in the [tfimsso:*jct name*] stanza in the WebSEAL configuration file.

For the second field, always set this value to asterisk (*)

(AppliesTo) Port Type

This property is requested on the Chain Mapping Lookup panel.

This property takes two fields.

Leave both fields blank.

(Issuer) Address

This required property is requested on the Chain Mapping Lookup panel. In the **Address** field, enter:

```
amwebrte-sts-client
```

(Issuer) Service Name

This optional property is requested on the Chain Mapping Lookup panel. Leave this field blank.

(Issuer) Port Type

This optional property is requested on the Chain Mapping Lookup panel. Leave this field blank

Token Type

This required property is requested on the Chain Mapping Lookup panel. Select **Kerberos GSS V5**.

Initialize the chain upon startup of runtime

This required property is requested on the Chain Identification panel. Do *not* select this option

Module Instances and modes

These required properties are requested on the Chain Assembly panel.

The Chain Assembly panel prompts you to enter values for the Module Instances in the chain. For each module instance, you must select a mode. You will then click a button to add the instance-mode pair to the chain.

For Kerberos constrained delegation, you want to configure a specific sequence of trust service modules:

1. The first Module Instance is **Default IVCred Token**. Choose a mode of **validate**.
2. The second Module Instance is the Kerberos delegation module instance that you created, as named in the **Module Instance Name** property within the module instance wizard. In our example, we used:

   ```
   MyKerberosDelegationInstance
   ```

   Select the **issue** mode.

Note: The wizard will warn you that your chain does not contain a module in **map** mode. For Kerberos constrained delegation, the map mode is not required.

You can add a map mode if your deployment requires it. A map module would be needed if the Tivoli Access Manager user name needs to be mapped to a different user name in the Active Directory registry.

In a typical deployment, this mapping is not required. For example, in many deployments, Tivoli Access Manager will be installed to use the Active Directory registry. In these cases, there is only one identity for each user.

Enable signature validation

This property is requested on the Access Manager Credential (IVCred) Module Configuration panel. Do *not* select this option.

Default target Service Principal Name

This property is requested on the Kerberos Delegation module configuration panel, as Partner property.

In a typical deployment, you can leave this value blank.

This value can be used for WS-Trust clients that do not send the target Service Principal Name (SPN) in the AppliesTo/ServiceName element of the RequestSecurityToken (RST). The clients would also not have a mapping rule to configure the target SPN as a security token service universal user (STSUU) context attribute.

Options for adding a Tivoli Access Manager username for Kerberos authentication

The options allow you to specify whether the module will auto-append a suffix to the user name in the STSUniversalUser. The options are useful when deploying the Kerberos delegation module with a Tivoli Access Manager WebSEAL deployment. Options:

- Do not add a suffix to the username.

  This option leaves the user name unmodified.
- Add the machine DNS domain as a suffix to the username.

  This option auto-appends the DNS domain suffix for the Tivoli Federated Identity Manager runtime machine to the principal name in the STSUniversalUser before calling the Windows API to obtain a Kerberos ticket. The DNS domain is read from the Windows Registry Key:

  ```
  SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Domain
  ```

  This option optimizes the module behavior for use in Tivoli Access Manager configurations using Kerberos junctions. The addition of the DNS domain enables the Windows API to successfully match the user name against the user record in the Active Directory user registry.

  Note that the module auto-appends the DNS domain name when the STSUniversalUser principal name does *not* already contain the @ character. This means that if a mapping rule was used to append a suffix containing the @ character to the user principal name, or if the Tivoli Access Manager username contains the @ character, this setting has no effect.
- Add the configured suffix to the username

  This option is used to optimize the module behavior for use in Tivoli Access Manager configurations using Kerberos junctions.

  This option allows the administrator to manually specify the suffix. This option is for special cases where the userPrincipalName attribute for the user does not match the DNS domain name of the Windows machine running the Tivoli Federated Identity Manager Runtime. This option has no effect when the principal name already contains an @ character.

  The suffix to add if using a configured suffix

  For example:

  ```
  @mydomain.com
  ```

Worksheet for trust chain configuration

Complete these worksheets prior to configuring the trust chain. The properties on the worksheets are described in “Planning configuration of the trust chain” on page 373.

Kerberos module instance worksheet

The following tables correspond to the panels presented by the module instance creation wizard.

Table 114. Module identification panels properties

Property	Value
Module type	com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule
Module Instance name	
Module Instance description	

Table 115. Kerberos Delegation Module Configuration panel property

Property	Your value
Maximum size of the user credential	Default: 100

Kerberos module trust chain worksheet

The trust chain wizard presents a series of configuration panels. The following tables correspond to each panel.

Table 116. Chain mapping identification properties

Property	Your value
Chain Mapping Name	
Chain Description	
Create a Dynamic Chain	*This option must be deselected*

Table 117. Chain Mapping Lookup properties

Property	Your value
Request Type	Issue Oasis URI
Lookup Type	Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)
(AppliesTo) Address	
(AppliesTo) Service Name	*Two fields* Use asterisk (*) for each field
(AppliesTo) Port Type	*Two fields* Leave both fields blank
(Issuer) Address	
(Issuer) Service Name	*Two fields* Leave both fields blank
(Issuer) Port Type	*Two fields* Leave both fields blank

Table 117. Chain Mapping Lookup properties (continued)

Property	**Your value**
Token Type	Kerberos GSS V5

Table 118. Chain identification panel

Property	**Your value**
Initialize the chain upon startup of runtime	*Do not select this option*

Table 119. Chain assembly panel

Property	**Your value**
First module instance	Default IVCred Token
First module mode	validate
Second module instance	*The name of your Kerberos module instance*:
Second module mode	Issue

Table 120. Access Manager Credential Module Configuration property

Property	**Your value**
Enable signature validation	*Deselect this option*

Table 121. Kerberos delegation module (Issue mode) Configuration property

Property	**Your value**
Default target Service Principal Name	
Options for adding a Tivoli Access Manager username for Kerberos authentication Options: • Do not add a suffix to the username. • Add the machine DNS domain as a suffix to the username. • Add the configured suffix to the username **The suffix to add if using a configured suffix** For example: `@mydomain.com`	

Creating a Kerberos constrained delegation module instance

About this task

A wizard guides you through the creation of the module instance. For information about each requested property, see “Planning configuration of the trust chain” on page 373.

You can also consult the “Worksheet for trust chain configuration” on page 376.

To create a module instance:

Procedure

1. Login to the WebSphere console.
2. Click **Tivoli Federated Identity Manager -> Configure Trust Service -> Module Instances** The Module Instances portlet is displayed.
3. Click **Create**. The Module Instance wizard starts, and the Module Type panel is displayed.
4. Select **com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule**. Click **Next**. The Module Instance Name panel is displayed.
5. Enter a value in the Module Instance Name field.

   For example:

   ```
   Kerberos Junction
   ```
6. Optionally, enter a description in the Module Instance Description field. .
7. Click **Next**. The Kerberos Delegation Module Configuration panel is displayed.
8. Enter a value in the field **Maximum size of the user credential cache**.
9. Click **Finish**. The Module Instances panel is displayed. The Current Domain portlet is also displayed, and prompts you to load the new configuration changes.
10. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.
11. Continue with “Creating a trust chain for Kerberos constrained delegation.”

Creating a trust chain for Kerberos constrained delegation

Before you begin

The domain must contain an instance of a Kerberos constrained delegation trust module before you build the trust chain. If you have not already created an instance, do so now. See “Creating a Kerberos constrained delegation module instance” on page 378.

About this task

To configure the trust chain correctly, you must ensure that the properties align with WebSEAL configuration properties. Before running the trust chain wizard, you should:

- Review the topic “Planning configuration of the trust chain” on page 373
- Complete the “Worksheet for trust chain configuration” on page 376

To build the trust chain:

Procedure

1. Login to the WebSphere console.
2. Click **Tivoli Federated Identity Manager -> Configure Trust Service -> Trust Service Chains** The Trust Service Chains portlet is displayed
3. Click **Create**. The configuration wizard starts. The Introduction page is displayed.
4. Click **Next**. The Chain Mapping Identification panel is displayed.

5. Enter the requested values.
   a. Enter a name in the **Chain Mapping Name** field.
   b. Optionally enter a description in the Description field.
   c. Do *not* select the field **Create a dynamic chain**
   d. Click **Next**. The **Chain Mapping Lookup** panel is displayed.
6. Enter the requested values.
   a. Set **Request Type** to **Issue Oasis URI**

      The corresponding value for Request Type URI is automatically entered by the wizard.
   b. Set **Lookup Type** to **Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)**.
   c. Enter values in the **AppliesTo** section.

      Enter values for the fields:
      - **Address**

        For example:

        ```
        http://websealhost.example.com/krbjct
        ```
      - **Service Name**.

        For example, set both fields to the asterisk character (*).
      - Leave the **Port Type** fields blank.

      For help, see "Planning configuration of the trust chain" on page 373
   d. Enter values in the **Issuer** section.
      - In the **Address** field, enter:

        ```
        amwebrte-sts-client
        ```
      - Leave the **Service Name** field and **Port Type** field blank.
   e. For **Token Type**, select **Kerberos GSS V5**
   f. Click **Next**.

      The Chain Identification panel is displayed.
7. Do *not* select **Initialize the chain upon startup of runtime**. Click **Next**.

   The **Chain Assembly** panel is displayed.
8. Build the trust chain:
   a. For Module Instance, select **Default IVCred Token**
   b. For Mode, select **validate**.
   c. Click **Add selected module to chain.**
   d. For Module Instance, select the Module Instance Name you specified in "Creating a Kerberos constrained delegation module instance" on page 378. For example,

      ```
      Kerberos Junction
      ```
   e. For Mode, select **issue**.
   f. Click **Add selected module to chain.**
9. Click **Next**.

   Note: You will see a warning stating that your chain lacks a module in map mode. You can ignore this warning. For more information, see "Planning configuration of the trust chain" on page 373.

   The Access Manager Credential (IVCred) Module Configuration panel is displayed.
10. Do *not* select **Enable signature validation**. Click **Next**.

The Kerberos Delegation Module Configuration panel is displayed.

11. If necessary, specify the Default target Service Principal Name or change the options for adding a suffix to the Tivoli Access Manager user name for Kerberos Authentication.

    Note: In most cases, you can leave this field blank and leave the default selection for the options. See "Planning configuration of the trust chain" on page 373
12. Click **Next**. The Summary panel is displayed.
13. Click **Finish**.
14. In the Current Domain portlet, click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

Results

The trust chain configuration is now complete.

Tivoli Federated Identity Manager configuration notes

Verify Tivoli Federated Identity Manager trust chain configuration

Verify that the WebSphere Deployment manager can communicate with the WebSphere Application Server that hosts Tivoli Federated Identity Manager.

To do this, access the URL:

```
http://<IHS_server>/TrustServerWST13/RequestSecurityToken
```

You will see a template response similar to the following:

```
RequestSecurityToken ... Hi there this is an AXIS service!
 Perhaps there will be a form for invoking the service here...
```

Verify WebSphere module mappings

Ensure that the WebSphere Application Server module mappings and virtual host mappings were propagated. To do this, access the URL:

```
http://<IHS_server>/Info/InfoService
```

You will see a template response similar to the following:

```
Hi there this is a Web service!
```

High availability in a cluster configuration

Multiple WAS servers will be deployed in a WAS cluster for high-availability. The individual WAS nodes in the cluster will receive their configuration instructions from a deployment manager.

Most administration tasks will be performed by communicating to the deployment manager. However, all of the protocol flows necessary to service requests to the TFIM trust service are served by individual WAS nodes. Failure of the deployment manager does not impact those protocol flows. Failure of the WAS nodes, however, will impact the protocol flow.

Chapter 33. WebSEAL configuration

You must install and configure the Tivoli Access Manager policy server before you install WebSEAL. These instructions assume that you have successfully installed and configured the policy server.

Complete a standard installation of WebSEAL. The exact steps to take depend upon your deployment environment. See the *IBM Tivoli Access Manager Installation Guide* for instructions.

Task overview:

1. "Verifying a WebSEAL installation"
2. "Planning WebSEAL Kerberos junction configuration" on page 384
3. Completing a "Kerberos junction configuration worksheet" on page 388
4. "Configuring a WebSEAL Kerberos junction" on page 389

Verifying a WebSEAL installation

Before you begin

These instructions assume that you have installed and configured IBM Tivoli Access Manager for e-business. The instructions also assume that you have successfully installed a WebSEAL server. This topic shows you how to verify that the basic WebSEAL server configuration is correct, so that you extend the configuration to support Kerberos junctions.

About this task

To verify the basic configuration, create a regular WebSEAL junction and verify that Tivoli Access Manager correctly prompts for a user login.

Complete the following steps:

Procedure

1. Obtain the WebSEAL server name.

   The server name is based on the host name. For example, with a host name of `webseal host`:

   ```
   pdadmin sec_master> server list
     default-webseald-websealhost
   ```
2. Create a simple junction.

   For example, when the protected server is `mydataserver`, the following command creates a junction at `/jct`:

   ```
   pdadmin sec_master> server task default-webseald-websealhost
    create -t tcp -h mydataserver/jct
   ```
3. Obtain the list of the /WebSEAL object.

   This value is needed in order to correctly attach an access control list (ACL):

   ```
   pdadmin sec_master> object list /WebSEAL
     /WebSEAL/websealhost-default
   ```
4. Attach an ACL to the new junction.

The ACL is used to control the actions that can be taken by specified users within the Tivoli Access Manager protected object space. This step assumes the existence of an ACL named `testacl`.

```
pdadmin sec_master> acl attach /WebSEAL/websealhost-default/testacl
```

5. To confirm that the junction and ACL are configured correctly, complete the following steps:
   a. Place a test file under the documentRoot on the protected Web server.

      For example, in the documentRoot for `mydataserver`, create a test directory and add an index.html that displays some content. For example, under the junction point, add the file:

      ```
      /testdir/index.html
      ```
   b. Access the protected content:

      ```
      https://websealhost.example.com/jct/testdir/index.html
      ```
   c. WebSEAL prompts you to log in. Log in with a valid Tivoli Access Manager user identity and password.

      When successful, you can view the contents of `testdir/index.html`.

Planning WebSEAL Kerberos junction configuration

WebSEAL configuration properties are specified in the WebSEAL configuration file. The default configuration file is webseald-default.conf. For example, on UNIX or Linux systems:

```
/opt/pdweb/etc/webseald-default.conf
```

The configuration file contains properties that support the deployment of Kerberos junctions. Before you can configure WebSEAL for Kerberos junctions, you must determine the values required by your deployment for each property.

The properties are grouped into two stanzas:

```
[tfimsso:jct-id]
[tfim-cluster:cluster]
```

In some cases, the introduction of a Kerberos single signon for junctioned servers can impact performance. Each Kerberos token is valid only for one Kerberos authentication. Therefore, WebSEAL must request a new Kerberos token for each separate transaction. Performance can also be impacted by the communication channel, which requires WebSEAL to obtain tokens through a SOAP request to Tivoli Federated Identity Manager.

[tfimsso:*jct_id*] stanza

[tfimsso:*jct_id*] stanza

Use the `[tfimsso:<jct-id>]` stanza to specify configuration options for using Kerberos single signon. This stanza contains the Tivoli Federated Identity Manager single sign-on configuration information for a single junction.

- For standard junctions, the stanza name must be qualified with the name of the junction point, including the leading forward slash. For example:

  ```
  [tfimsso:/kerbjct]
  ```
- For virtual host junctions, the stanza name must be qualified with the virtual host label, for example:

```
[tfimsso:www.example.com]
```

always-send-tokens

Boolean property. This property can be used to optimize performance when the back-end (junctioned) server is capable of maintaining session state. In this case, you can specify whether WebSEAL should send a Kerberos token for every HTTP request, or if WebSEAL should wait for a 401 response before requesting the token.

A 401 response means that authorization is required. When session state is maintained, it not necessary to authorize prior to each request. To limit the retrieval of Kerberos tokens to only those times when authorization is required, set

```
always-send-tokens = false
```

When the backend server cannot maintain session state, and a security token should be sent for every HTTP request, set:

```
always-send-tokens = true
```

applies-to

This property specifies the search criteria to use when locating the correct security token service module within Tivoli Federated Identity Manager.

The value is typically a path consisting of the format:

```
http://webseal_server_host/junction_name
```

For example:

```
http://webseahost.example.com/kerbjct
```

service-name

This important property is used for two purposes:

1. To specify the service principal name that is used when generating a Kerberos token. This value is used by Tivoli Federated Identity Manager when it searches for a matching trust chain. The Tivoli Federated Identity Manager chain configuration includes an Applies-to section that contains a Service Name property. The value of the WebSEAL service-name setting is compared against the Service Name property.

   To ensure a successful match, `service-name` should match the Service Name property in the Tivoli Federated Identity Manager configuration.

   Note: One way to ensure a successful match is to use, within the Tivoli Federated Identity Manager configuration, a wildcard character such as asterisk (*).
2. To specify the service principal name of the delegating user when creating the Kerberos token. The service principal name (SPN) is set on the Microsoft Windows system.

   To determine the SPN, go to the Windows server, and use the **setspn** command. For example:

   ```
   setspn -L user_name
   ```

   The junctioned Web server runs with the identity *`user_name`*. For example, iisuser.

The syntax for this property is:

```
service-name=service_principal_name
```

The format is:

```
HTTP/IIS_server_name.domain_name
```

For examples:`service-name = HTTP/B16INTEL3.tamad.com`

renewal-window
The length of time, in seconds, by which the expiry time of a security token is reduced. This entry is used to accommodate differences between system times, and to allow for transmission times for the security tokens.

```
renewal-window = 15
```

tfim-cluster-name
The name of the WebSphere cluster where the Tivoli Federated Identity Manager service is deployed. This value should be matched by another stanza entry [tfim-cluster:<*cluster*>], where *cluster* matches **tfim-cluster-name**.

For example:

```
tfim-cluster-name = STSCluster2
```

token-collection-size
To optimize performance, WebSEAL can request multiple Kerberos tokens from Tivoli Federated Identity Manager within one SOAP request. This is done through use of the WS-Trust Web service specification. The tokens are cached in the user's session and used on subsequent requests. WebSEAL requests additional tokens from Tivoli Federated Identity Manager only after all of the cached tokens have been used or have expired.

You can specify the number of tokens to retrieve from Tivoli Federated Identity Manager. When this number is increased, the number of requests to Tivoli Federated Identity Manager is decreased, but the size of (and processing time for) each request is increased. The Kerberos tokens can be quite large. If you specify a large value for this property, you can significantly increase the session size and memory usage for WebSEAL.

The default value is 10:

```
token-collection-size = 10
```

token-type
The only supported token type is `kerberos`. This is the default value. Use this value. Do not change it.

tfim-cluster cluster stanza

[tfim-cluster:*cluster*]

This value defines the name of the WebSphere cluster for the Tivoli Federated Identity Manager service. The *cluster* name for this stanza must match the **tfim-cluster-name** option in a **[tfimsso:*jct-id*]** stanza.

server Specifies the priority level and URL for a single Tivoli Federated Identity Manager server that is a member of the cluster identified for this stanza.

You can have multiple **server** entries in the stanza. This enables you to specify multiple server entries for failover and load balancing purposes between WebSEAL and the WebSphere Application Server proxy. When the Tivoli Federated Identity Manager cluster is configured, WebSEAL checks the status of the Tivoli Federated Identity Manager proxy Web server once every minute.

When you have multiple servers, you can use the priority level to specify the order in which the servers are accessed to perform processing. The priority level is an integer in the range [0-9].

When you have only one server, you can omit the priority level. When the priority level is not specified, the level is assumed to be 9 (highest).

Syntax:

```
server = [0-9],server_URL
```

Example:

```
9,http://mydataserver.example.com/TrustServerWST13/services
/RequestSecurityToken
```

handle-pool-size

Specifies the maximum number of cached handles to use when communicating with Tivoli Federated Identity Manager.

Default: 10

handle-idle-timeout

The length of time, in seconds, before an idle handle is removed from the handle pool cache

Default: 240 seconds

timeout

The length of time, in seconds, to wait for a response from Tivoli Federated Identity Manager.

Default: 240 seconds

ssl-keyfile

The name of the key database file which houses the client certificate to be used.

This SSL entries, and the ones following, are optional and are only required when:

- At least one server entry indicates that SSL (HTTPS) is to be used.
- A certificate is required other than that which is used by this server when communicating with the policy server

Note: This value, and the following SSL entry values, must be shared for all server variables that use HTTPS. When deploying into a WebSphere cluster, the values must be the same for each server in the cluster that uses HTTPS.

ssl-keyfile-stash

The name of the password stash file for the key database file.

ssl-keyfile-label

The label of the client certificate within the key database.

ssl-valid-server-dn

This configuration entry specifies the DN of the server (obtained from the server SSL certificate) which will be accepted. When no entry is configured, all DN's will be considered to be valid. Multiple DN's can be specified by including multiple configuration entries of this name.

ssl-fips-enabled

This entry controls whether FIPS communication is enabled with Tivoli

Federated Identity Manager or not. When no configuration entry is present, the global FIPS setting, as determined by the TAM policy server, will take effect.

Note: For a complete description of each stanza property, see the *IBM Tivoli Access Manager WebSEAL Administration Guide*. See also the comments within the WebSEAL configuration file.

Kerberos junction configuration worksheet

Use this worksheet to assemble the values that you must add to the WebSEAL configuration file.

Table 122. tfimsso and tfim-cluster stanza properties

Property	Your value
[tfimsso:*junction_id*]	
always-send-tokens	default: false
applies-to	
service-name	
renewal-window	default: 15
tfim-cluster-name	
token-collection-size	default: 10
token-type	kerberos
[tfim-cluster:*cluster*]	
server	
handle-pool-size	default: 10
handle-idle-timeout	default: 240
timeout	default: 240
ssl-keyfile	
ssl-keyfile-stash	
ssl-keyfile-label	
ssl-valid-server-dn	
ssl-fips-enabled	

Configuration tips:

- Ensure that the **service-name** property matches the Tivoli Federated Identity Manager trust chain configuration.
- Ensure that the **tfim-cluster-name** property matches the *cluster* property in the stanza [tfim-cluster:*cluster*].
- Ensure that the *cluster* property in [tfim-cluster:*cluster*] matches the name of the WebSphere cluster.

Configuring a WebSEAL Kerberos junction

About this task

Configuration of a WebSEAL Kerberos junction consists of two tasks:

- Edit the WebSEAL configuration file

  You must specify properties in the WebSEAL configuration file to support the specific junctions for Kerberos single-signon before you can use the pdadmin command to create the junction.
- Use the pdadmin command to create the junction and attach the necessary ACLs.

  To create a standard junction that is enabled for Kerberos single signon, use the junction create command (server task create) with option -Y. The -Y option specifies that SPNEGO/Kerberos single sign-on is required for the junction.

  To create a virtual host junction that is enabled for Kerberos single signon, use the virtualhost create command (server task create) with the -Y option.

  WebSEAL supports many options for creating junctions. You can combine the -Y option with other options, as required for your deployment. For complete information on WebSEAL junction options, see the *IBM Tivoli Access Manager WebSEAL Administration Guide*.

To configure a WebSEAL Kerberos junction:

Procedure

1. Use a text editor to edit the WebSEAL configuration file

   Use the values that you assembled in the worksheet for Kerberos junction support.

   For more information, see “Planning WebSEAL Kerberos junction configuration” on page 384
2. Use the pdadmin command to create the Kerberos junction and attach the necessary access control lists (ACLs).

   You can create either regular Kerberos junctions or virtual host Kerberos junctions.

   Note:
   - The name of the junction must match the *jct_id* value for the [tfimsso:*jct_id*] stanza in the WebSEAL configuration file.
   - Ensure that you have configured the WebSEAL configuration file for the type of junction that you want to use. If you have not edited the WebSEAL configuration file, the administration command will not succeed, and will return an error message.

   Regular Kerberos junctions

   a. Create the junction:

   ```
   pdadmin sec_master>  server task default-webseald-websealhost
     create -t tcp -h mydataserver.example.com -Y /kerbjct
   ```

      The host mydataserver.example.com is the IIS backend server.

   b. Attach the ACL:

   ```
   pdadmin sec_master> acl attach /WebSEAL/websealhost-default/kerbjct testacl
   ```

   Virtual host Kerberos junctions

   a. Create the junction:

```
pdadmin sec_master> server task default-webseald-websealhost virtualhost
  create -t tcp -h mydataserver.example.com -v website.example.com
  -Y kerbvirtjct
```

b. Attach the ACL:

```
pdadmin sec_master> acl attach /WebSEAL/websealhost-default/kerbvirtjct
  testacl
```

Results

Error messages are logged in the WebSEAL configuration log file. For example, on UNIX or Linux:

```
/opt/pdweb/log/msg__webseald-default.log
```

WebSEAL configuration notes

Configuration notes for communication between WebSEAL and the client

- High availability for the WebSEAL server is typically done by placing a load-balancer in front of the WebSEAL server. See the IBM Developer Works article *Load Balancers for Tivoli Access Manager*:

  http://www-128.ibm.com/developerworks/tivoli/library/t-tlb/index.html
- Security of the communication path between the client and WebSEAL is typically provided by purchasing an SSL server certificate for the WebSEAL server.
- Clients may authenticate to WebSEAL through any supported method.
- No change to these standard configurations will be necessary for Kerberos junction support.

Configuration notes for communication between WebSEAL and the junction

- High availability for the junctioned server is typically done by configuring multiple junction servers for the junction point. See the IBM Developer Works article *Load Balancers for Tivoli Access Manager*:

  http://www-128.ibm.com/developerworks/tivoli/library/t-tlb/index.html
- Security of the communication path between WebSEAL and the junction is typically guaranteed by mutually authenticated SSL certificates.
- No change to these standard configurations is required for Kerberos junction support.

Time synchronization between WebSphere and WebSEAL

Verify that time settings are synchronized between the system that hosts the WebSphere Application Server that runs Tivoli Federated Identity Manager and the system that hosts Tivoli Access Manager WebSEAL.

To view the settings:

1. On the WebSphere system, navigate to Default Domain Security Settings, Account Policies, and Kerberos Policy.
2. Review the Maximum tolerance for computer clock synchronization.

When the time difference is large between the WebSphere Application Server and the WebSEAL server, the security tokens generated by Tivoli Federated Identity Manager might expire before they can be used.

Configuration error messages

The following error messages are displayed when one of the following conditions are true:

- The service-name property does not match the Tivoli Federated Identity Manager trust chain configuration
- WebSEAL retrieves tokens from Tivoli Federated Identity Manager, but the tokens have expired. This can happen, for example, when the time settings on each of the servers are not synchronized.
- The browser returns an error. For example:

  ```
  Server Error
  Access Manager WebSEAL could not complete your request due to an
  unexpected error.
  Diagnostic Information
  Method: GET
  URL: /kjct/index.html
  Error Code: 0x38cf027c
  Error Text: DPWWA0636E No TFIM single sign-on tokens were available.
  ```
- The WebSEAL log contains errors. For example (some lines split for formatting purposes):

  ```
  DPWWA2852E   An error occurred when attempting to communicate with the SOAP
    server URL
  http://d06win13.testlab.example.com/TrustServerWST13/services/
    RequestSecurityToken: +JNI:
  Error running InitializeSecurityContext for HTTP/d02jlnx.testlab.example.com:
  -2146893042 (No credentials are available in the security package).
  File h:\fim620\src\kerberoswin32\KerbUserState.cpp,
   line 641  (error code: 71/0x47).
  2008-03-04-13:08:10.080-06:00I----- 0x38CF027C
  webseald ERROR wwa sso ThirdPartyJunction.cpp 4124 0x00000070
  DPWWA0636E   No TFIM single sign-on tokens were available.
  ```

Debugging a Kerberos junction

To debug a Kerberos junction deployment, turn on tracing for Tivoli Federated Identity Manager and Tivoli Access Manager. A relevant trace point for Tivoli Access Manager and WebSEAL is pdweb.sso.tfim.

For example, in a Linux or UNIX environment:

```
pdadmin> server task default-webseald-c1sun1 trace set pdweb.sso.tfim 9
  file path=/var/pdweb/log/debug.log
```

Set the trace level to 0 to turn off tracing.

Configuring WebSEAL to manage cookies

By default, WebSEAL does not delete cookies upon logout. If you plan to configure WebSEAL to manage cookies, the list of managed cookies should not include the WebSphere session cookie.

Chapter 34. SSL configuration task for a Kerberos junctions deployment

For optimal security, configure SSL communication between servers in a Kerberos junction deployment.

This topic provides an overview of the steps to configure a WebSphere cluster environment to use SSL to communicate between WebSEAL, IBM HTTP Server (IHS), WebSphere Application Server Plug-in, WebSphere Application Server and Tivoli Federated Identity Manager. These steps do not address SSL communication between the client and WebSEAL or to the back-end Web server. No changes to these standard SSL configurations are necessary for Kerberos junction support.

Tip: Consider deploying a working configuration without SSL prior to adding SSL.

For each component, create a public/private key pair, and extract the public key to a known location.

On the WebSEAL server:

1. Copy the IHS public key to the WebSEAL system
2. Use the **ikeyman** utility to add the IHS public key. When there is more then one IHS proxy in the environment, complete this task for each IHS server.
3. Configure appropriate values for the following `[tfim-cluster:cluster]` variables: server, ssl-keyfile, ssl-keyfile-stash. Optionally configure the ssl-valid-server-dn variable if applicable.

   For more information, see “Planning WebSEAL Kerberos junction configuration” on page 384.
4. Restart WebSEAL to activate the changes made to the WebSEAL configuration file.

On the IBM HTTP Server:

1. Copy the WebSEAL public key to the IHS system.
2. Use the ikeyman utility on IHS to add the WebSEAL public key.
3. Copy the WebSphere public key from the WebSphere Deployment Manager (dmgr) system to the IHS system.
4. Use the ikeyman utility on IHS to add the WebSphere public key.
5. Update the httpd.conf file to configure or add a virtual host to support SSL connections.
6. Restart IHS to activate the changes.
7. When your deployment includes multiple IHS proxies, repeat the above steps for each IHS proxy.

On the WebSphere plug-in located on the IHS server:

1. Copy the WebSphere public key to the plug-in system.
2. Use the ikeyman utility for the plug-in to add the WebSphere public key.
3. Copy the WebSphere node public key from the WebSphere node to the plug-in server.
4. Use the ikeyman utility for the plug-in to add the WebSphere node public key.

5. When your deployment includes multiple plug-ins, repeat the above steps for each plug-in.

On the WebSphere Network Deployment Manager (dmgr):

1. Ensure that the public key for the plug-in is located in a file path that can be accessed through the WebSphere administration console.
2. Use the WebSphere console to add the public key for the plug-in to the CellDefaultTrustStore.
3. When your deployment includes multiple plug-ins, repeat the above steps for each plug-in.
4. Ensure that the public key for Node is located in a file path that can be accessed through the WebSphere administration console.
5. Use the WebSphere console to add the public key for the Node to the CellDefaultTrustStore.
6. When your deployment includes multiple nodes, repeat the above steps for each nodes.
7. Configure client authentication if appropriate for your deployment.

On the WebSphere Node:

1. Ensure that the public key for the Deployment Manager (dmgr) is located in a file path that can be accessed through the WebSphere administration console.
2. Use the WebSphere console to add the dmgr public key to the NodeDefaultTrustStore.
3. When your deployment includes multiple nodes, repeat the above steps for each nodes.

Part 5. Configuring User Self Care

The topics in the Configuration section provide a step-by-step guide to configuring User Self Care.

This section describes the deployment of User Self Care. First read the overview of the User Self Care feature:

Chapter 35, “Understanding User Self Care,” on page 397

Chapter 35. Understanding User Self Care

User Self Care provides a method by which users can be provisioned into business-to-consumer environments. User Self Care accomplishes this provisioning by supplying a set of operations that users can use to create and administer their own accounts. The operations include:

- Creating an account
- Creating and updating attributes associated with the account
- Changing passwords
- Recovering forgotten user IDs and passwords
- Deleting accounts

User Self Care is based upon the Tivoli Federated Identity Manager secure token service (STS) technology. With the STS framework, administrators can plug in their own token creation and consumption modules. User Self Care uses the STS framework and the HTTP components of Tivoli Federated Identity Manager, but it is not used for token creation and consumption.

Users access User Self Care operations through an HTTP interface. Users interact with web pages that prompt for input, collect data, and provide feedback. User Self Care provides a small set of URLs that serve as endpoints for accessing operations.

You can customize User Self Care. STS modules plug-ins that are started sequentially in a chain implement business logic. To provide additional capability for each chain, you can replace individual modules or add new ones. You can modify or replace the HTML forms as necessary.

User Self Care uses the clustering, distribution, scaling, and configuration capabilities provided by WebSphere. User Self Care also uses the WebSphere Federated Repositories component for making registry adapters available to the operating environment. Administrators can add or replace registries.

User Self Care also integrates with Tivoli Access Manager WebSEAL. WebSEAL provides authentication and authorization for business-to-consumer transactions.

The figure shows the software pieces that comprise the User Self Care solution.

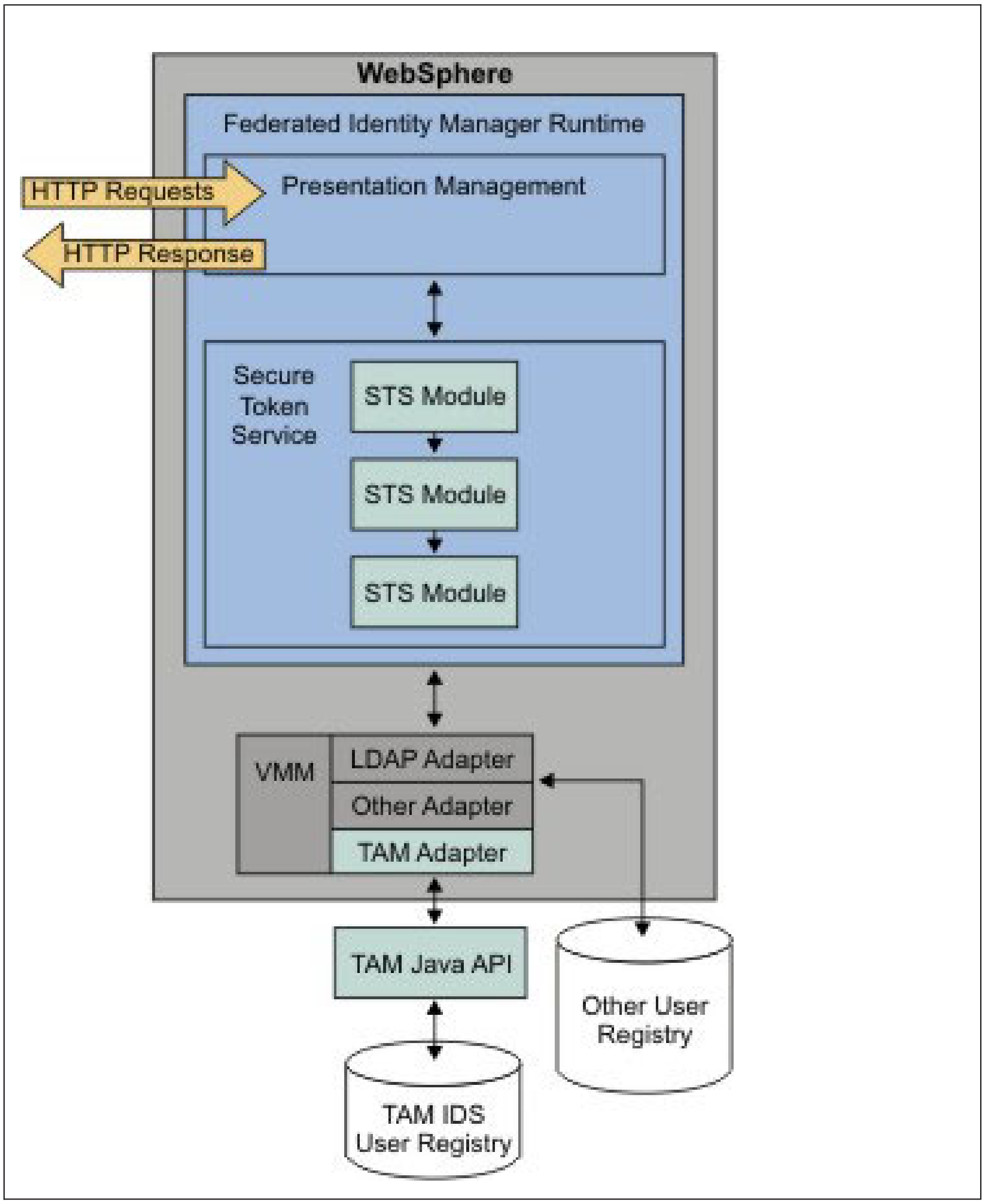


Figure 51. User Self Care solution

- WebSphere provides the framework for most of the software pieces.
- The Tivoli Federated Identity Manager run time provides two components that support User Self Care:

  User Self Care presentation management
  Provides a set of default pages. Users interact with these pages by requesting User Self Care URLs. The management framework supports customization and replacement of these pages. This support includes the ability to substitute (customize) macros on the pages.

  Secure token service (STS) trust chains
  Supports the building of dynamic chains of plug-in modules for performing business logic. User Self Care support includes a number of STS chains. Each chain maps to a User Self Care operation. You can extend the chains. You can replace or modify the component modules in

each chain. Validating user input and sending a confirmation e-mail is an example of a User Self Care chain operation.

- The STS modules use the WebSphere Federated Repository to communicate with the user registry. When the target user registry is Tivoli Access Manager, User Self Care uses a Tivoli Access Manager adapter to communicate to the Tivoli Access Manager registry through the Tivoli Access Manager Registry Direct Java API.

User Self Care works with various user registries. Each registry has a unique syntax for performing management operations. The WebSphere Federated Repositories component permits User Self Care to issue a management command, such as **user create**, using a consistent syntax. The Federated Repositories component then passes the request to the appropriate registry adapter, which translates the command into the registry-specific syntax. Since WebSphere Federated Repositories provides a plug-in interface for adapters, you can add new registries without modifying the User Self Care.

Effectively customizing User Self Care

Deployments of User Self Care are generally customized for specific business needs. You can most effectively customize your deployment when you understand how the User Self Care pieces work together.

1. Understand the User Self Care technology.
   - User Self Care is based on a series of operations. See “Understanding User Self Care operations.”
   - Users interact with User Self Care features through HTTP request and response exchanges. HTML pages as URL drive the exchanges The default HTML pages are templates for the information you want to exchange with your users. You can (and should) customize the HTML pages to reflect your business needs. For more information on default HTML pages, see “User Self Care URLs” on page 407.
   - Many Internet sites use Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) challenge-response tests to protect against machine-generated attacks. This technology is part of many User Self Care deployments. The User Self Care product provides a Captcha demonstration module. See “Captcha demonstration” on page 410
2. Deploy Tivoli Federated Identity Manager and configure User Self Care.

   This document provides configuration steps that you must do in a specific order. See Chapter 36, “Deploying User Self Care,” on page 413.
3. Understand the methods for tuning distributed caches to optimize performance. See Chapter 37, “Tuning User Self Care,” on page 433.

Understanding User Self Care operations

A User Self Care *operation* is the series of steps required for a user to accomplish a task, such as a user recovering a forgotten password. To perform this task, the user must take several steps:

1. Submit a web form with their user ID
2. Submit a second form that asks them to answer their secret question and to provide a new password
3. Click a link in an e-mail that is sent to them.

The combination of steps comprise an *operation*.

Every user-initiated action performed as part of User Self Care is in the form of an HTTP request. Example requests are requesting a page, submitting a form, or clicking a link in an e-mail. Every HTTP request has a corresponding HTTP response. Some example responses are providing the user with the input form or informing them that an e-mail has been sent. Each User Self Care operation consists of one or more request-response exchanges.

In most cases, each request-response exchange is an atomic event. For example, when a user requests the Profile Management page, User Self Care finishes a discrete operation by returning the page. User Self Care does not retain any state or knowledge that the user has made the initial request. There are exceptions to this treatment of user state, which are described in the individual operation topics of this documentation.

This documentation groups the request-response exchanges based on their association with an operation. Each operation is associated with a particular secure token service trust chain. The STS trust chains do the bulk of the work in processing a User Self Care operation.

A typical request-response exchange

The typical flow when a user submits a request to User Self Care is:

1. User requests a User Self Care URL that specifies an HTML form.
2. The User Self Care presentation management component returns the appropriate HTML form.

   If the Captcha module is used, the Captcha STS chain is started to obtain the image shown to the user for validation.
3. User supplies data for the form and submits it.
4. The presentation management component sends the resulting HTTP request to the appropriate STS trust chain.
5. The User Self Care STS trust modules in the chain are started in a specified order, to perform tasks such as:
   - Validating data
   - Mapping attribute
   - Interacting with registries
   - Sending e-mail
6. The STS modules return the process results to the presentation management component.
7. The presentation management component returns an HTTP response to the user. The response typically is one of the following:
   - Another form
   - The same form with a message
   - An error page
   - An informational page

   Depending upon the operation, the user task is either finished or requires another step. If another step is required , the preceding or similar sequence is repeated.

Operations and STS chains

Each user self care operation maps to a single STS chain. During the operation, the STS chain might be started multiple times. User Self Care determines what stage of the operation is being performed and controls behavior accordingly.

For example, Captcha validation might be performed when a user submits the initial enrollment form. However, it is not performed when the user clicks the link in the e-mail. In both cases, the same STS chain is started, and the Captcha STS module is present at the start of the chain. In the second case, the Captcha module is not supposed to do anything, and passes the request to the next STS module in the chain.

You can use the administration console to view each trust chain. Trust chains correspond to one or more User Self Care operations. When you view the trust chains, you see the STS modules that accomplish the operation. You can then customize the modules and chains for your deployment.

Note: For information about how to customize User Self Care, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis/display/tivolifederatedidentitymanager/Home

User ID existence check operation

On the initial enrollment page, the user enters a user ID in a specified field. User Self Care provides an icon that the user can click to check if the ID exists in the registry. The user ID existence operation is an exception to the rule of one STS chain per operation. This operation maps to the same STS trust chain as the enrollment operation. However, it is conceptually different and uses a different URL.

Operation task flow:

1. User enters their requested user ID in a form field.
2. User clicks the icon.
3. The Create Account STS Chain is started.
   - The registry is queried to determine whether the user ID exists.
   - The internal cache is also queried.

     The check of the internal cache is described in “Enrollment operation.”

Enrollment operation

The enrollment operation takes place in two request-response exchanges:

1. Obtain user information in preparation for sending a validation e-mail.
2. The user validates the operation by clicking a link in the e-mail.

Initial enrollment request

Operation task flow:

1. User requests and receives an Enrollment Request form. User supplies data for the form fields with enrollment details such as:
   - User ID
   - E-mail address

- Password
- Choice of profile attributes, including the secret question attribute.

2. User submits the Enrollment Request form.
3. The Create Account STS Chain is started.
   - If any errors are encountered, they are returned to the user. The errors are shown as a message on the form that the user initially processed.
   - If no errors are encountered, an e-mail is sent to the user for validation. User Self Care shows a page to the user, advising them of the e-mail.
4. An entry is created in an internal cache that preserves the user enrollment information during the validation. This internal cache also preserves the user ID so that no other user can use it for enrollment. You can configure the time limits for how long data is retained in the internal cache.

Enrollment validation

The e-mail that was sent during the initial enrollment request contains a link with a query string appended. The query string contains a key to the internal cache entry so that the data that the user initially submitted can be recovered and enrollment finished.

Task flow:

1. User clicks a link in the validation e-mail.
2. The Create Account STS Chain is started.
3. If any errors are encountered, they are shown in a page that is sent to the user. If no errors are encountered, User Self Care:
   a. Creates an entry in the registry for the new user account
   b. Removes the internal cache entry
   c. Sends a success message to the user.

Password management operations

There are two password management operations:

- A user-initiated change of password
- A password change required by expiration of an existing password

User-initiated change password

Task flow:

1. The user requests the Change Password Form URL.
2. User Self Care provides the user with a form in which they enter their old password and a new password twice.
3. The user submits the form to the Change Password URL.
4. User Self Care starts the Change Password STS Chain.
   - If any errors are encountered, User Self Care sends the user on an informational page containing the errors.
   - If no errors are encountered, the password is changed. User self care then sends a success page to the user.

Change password following password expiration

The task flow is the same as in the User-initiated change password topic, with the exceptions that:

- The user makes an initial request for a protected resource
- The point of contact server requires the user to change their password.

The initial request from the user is intercepted by the authenticating point of contact server, such as WebSEAL or WebSphere Application Server. The point of contact server handles the communications flow and must direct the user to User Self Care in order to change their password. User self care provides deployment suggestions and enhancements for accomplishing this using WebSEAL as a point of contact server. For more information, see “Integrating User Self Care with WebSEAL” on page 427.

User self care can function as a callable component for a capability such as the Tivoli Access Manager Local Response Redirect feature. This feature redirects the user to the User Self Care handler to perform a change password operation. The user is then redirected back after the operation succeeds.

Profile management operations

Profile management permits a user to manage extended information specific to their account. Examples of such information are:

- Address
- Phone number
- Secret question

Initial Profile Management Request

Task flow:

1. User submits request for the Profile Management Form URL. This URL must be a protected resource.
2. The user identity is obtained from the authenticated context
3. User Self Care starts the profile management STS Chain, and provides the user identity.
   - If errors are encountered, they are shown in an informational page that is sent to the user.
   - If no errors are encountered, the STS retrieves the attributes from the registry.
4. User Self Care presents the user with the Profile Management Form containing their existing attributes. The user can then update profile information, including their secret question.

Submit Profile Update

Task flow:

1. User modifies the wanted fields and submits the form.
2. The user identity is obtained from the authenticated context.
3. User Self Care starts the profile management STS Chain.
   - If errors are encountered, they are shown in an informational page that is sent to the user.

- If no errors are encountered, the registry is updated. User self care sends a success page to the user.

Forgotten user ID operation

Operation task flow:

1. User clicks on the Forgotten ID URL. This URL must not be a protected resource.
2. The Forgotten ID form is returned to the user.
3. User enters their e-mail address.

   A custom solution can use a different registry attribute, such as a customer account number, for example. The default User Self Care form uses the e-mail address.
4. User submits the form.
5. User Self Care passed the form contents to the Forgotten ID STS Chain. The modules in this chain retrieve all the user IDs associated with the e-mail address from the registry, and then e-mail them to the user.
   - If errors are encountered, they are shown in an informational page sent to the user.
   - If no errors are encountered, User Self Care sends the Forgotten ID Acknowledgement informational page to the user. The page informs the user that the user IDs have been sent to their e-mail address.

Forgotten Password operation

The forgotten password operation takes place in several request-response exchanges.

Task flow:

1. User requests the Forgotten Password URL. This URL must not be a protected resource
2. User Self Care sends the Forgotten Password Form to the user.
3. User enters their user ID and submits the form.
4. User Self Care passes the form contents to the Forgotten ID STS Chain to retrieve the secret question.
5. The STS module sends the Forgotten Password Secret Question Form to the user. The form has the secret question and a field in which the user must enter the answer. The form also provides two fields for capturing a new password.
6. The user edits and submits the Secret Question form.
7. User Self Care passes the form contents to the Forgotten ID STS Chain to perform Secret Question Validation.
   a. The STS tracks the number of failed attempts in an internal cache. If the number exceeds the configured limit, the STS sends an error to the user.
   b. The STS stores the password change request in an internal cache.
   c. The STS sends an e-mail to the user containing a link to the Forgotten Password Validation Form URL. The e-mail contains a link with a query string appended. The query string contains a key to the internal cache entry. The key is used so that the data that the user submitted can be recovered and the password change finished.
8. User requests the link in the e-mail.

9. User Self Care passes the request to the Forgotten ID STS Chain. The chain modules recover the data from the internal cache and attempt to change the password.
   - If errors occur, they are shown in an informational page sent to the user.
   - If no errors occur, User Self Care sends to the user the Forgotten Password Acknowledgement informational page. This page tells them that their password has been changed.

Account deletion operation

Operation task flow:

- The user requests the Account Deletion page. This page must be a protected resource.
- The user clicks a link on the page.
- The user identity is obtained from the authenticated context.
- The Account Deletion STS chain is started.
- The Account Deletion STS trust chain finishes the deletion of the user account.
- User Self Care returns the Account Deletion success informational page to the user.

Captcha operation

Captcha is not a separate User Self Care operation. Instead, the Captcha operation is implemented as a Captcha STS module. You can place the module first in any of the secure token service trust chains used by User Self Care. When the Captcha module is present, Captcha validation is performed before execution of any other operations.

For more information, see "Captcha demonstration" on page 410.

Registry attributes operations

User Self Care does not provide the capability to modify user registry schema. You must modify your registry schema as required to create the registry attributes required for supporting profiles. You must also modify the schema to support the *secret question* attribute.

User Self Care provides an example function. User Self Care uses the LDAP attribute `businessCategory` to store the secret question profile attribute. The example implementation also uses the LDAP attribute `mobile` to store a mobile phone number for the user.

When you deploy User Self Care, you must create a schema that can contain the profile attributes you must provide for your users. When you have identified and defined these attributes, you must customize the HTML forms and STS modules to work with them.

In a full deployment, it is necessary to create a schema that can contain the profile attributes you must provide for your users. When these attributes are selected, you must customize the HTML forms and the STS modules in order to work with the new attributes.

For more information, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis%2Fdisplay%2Ftivolifederatedidentitymanager%2Fhome.

Secret question operation

The *secret question* is a secondary password and hint stored in the user registry as a user attribute. User Self Care treats the management of the secret question as another profile element.

User Self Care provides an example implementation of the secret question by using the LDAP attribute businessCategory to store the secret question profile. You can customize this implementation to best fit your business needs.

The following topics describe how the example implementation works.

Selection of secret question during enrollment

A User Self Care enrollment form provides a menu that permits a user to select one of the following questions:

- Maiden name of mother
- Town where you were born
- Name of first pet

The selection of one of these items populates a form field with a numeric value that corresponds to the index of the entry in the list. The name of this field on the HTML forms provided with user self care is `usc.form.profile.secret.question`.

A separate form field is used to specify the answer to the question in text. The name of this attribute on the HTML forms provided with User Self Care is `usc.form.profile.secret.question.answer`.

When the user submits the enrollment form, each of these parameters is passed to the Enrollment STS trust chain. The index and the answer are concatenated together and stored in the LDAP attribute `businessCategory`.

Showing the secret question during profile management

When the user requests the profile management form, User Self Care retrieves the user attributes, including the secret question, from the registry. The profile management STS module parses the attribute and determines the index specifying the secret question that the user has previously selected. User Self Care then uses this index value to show the appropriate value from the menu in the profile management form.

Using the secret question to validate the user identity

When the user submits the forgotten password form, User Self Care uses the user ID to retrieve the `businessCategory` registry attribute. The Forgotten Password STS module then parses the value of the attribute and returns the index to the presentation management component. This component uses the index to perform a macro substitution. The substitution provides a value to JavaScript that drives the selection of the matching secret question.

Secret question implementation tip

The security of the secret question approach is improved when users can create their own secret question. The convenience offered by a menu list is more than offset by the risk of providing pieces of identifying information. The information is often reused across many Internet sites.

The default values provided by User Self Care are for example use only. They include commonly used values such as maiden name of mother, favorite color, and name of first pet. As a preferred security practice, do not use these values in an enterprise deployment.

User Self Care URLs

User Self Care provides a set of default HTML pages for communicating with the user. The HTML pages facilitate the exchange of HTTP requests and responses.

- "User Self Care HTTP requests"
- "User Self Care HTTP responses" on page 409

User Self Care HTTP requests

The following table lists the URLs that are requested by users when interacting with User Self Care. Some URLs are listed for more than one request. Each URL is unique to a User Self Care operation and maps to an STS chain. User Self Care determines what phase of the operation is performed by examining the contents of the request.

Note: User authentication is required for some URLs. If the description does not mention user authentication, no authentication is required.

Table 123. HTTP Requests

Name	HTTP Method	Request URI and Description
Master Page	GET	Optional custom page not hosted by User Self Care. You might want to create a page that contains links to User Self Care operations but is not hosted by User Self Care.
Enrollment Request Form	GET	/sps/*federation_name*/usc/self/account/create Requests the enrollment form.
Enrollment Request Submit	POST	/sps/*federation_name*/usc/self/account/create Submits the enrollment form.
Retrieve user ID	POST	/sps/*federation_name*/usc/global/userid/search Maps to a separate User Self Care operation that determines if a user ID exists. This page results from clicking a link on the Enrollment Request Form.
Enrollment Validation	POST	/sps/*federation_name*/usc/self/account/create/validate Specifies the base URL in the e-mail sent to the user during enrollment validation. The final URL has a query string appended to it.

Table 123. HTTP Requests (continued)

Name	HTTP Method	Request URI and Description
Change Password Form	GET	/sps/*federation_name*/usc/self/password/update Authentication required Requests the change password form.
Change Password Submit	POST	/sps/*federation_name*/usc/self/password/update Authentication required Submits the change password form.
Forgotten ID Form	GET	/sps/*federation_name*/usc/self/account/recover/userid Requests the forgotten ID form.
Forgotten ID Submit	POST	/sps/*federation_name*/usc/self/account/recover/userid Submits the forgotten ID form.
Forgotten password Form	GET	/sps/*federation_name*/usc/self/account/recover/password Requests the forgotten password form.
Forgotten password Form	POST	/sps/*federation_name*/usc/self/account/recover/password Submits the forgotten password form.
Forgotten Password Secret Question Form	POST	/sps/*federation_name*/usc/self/account/recover/password/secretquestion Submits the secret question validation form. This form is presented to the user after they submit the forgotten password form.
Forgotten Password Validation Form	POST	/sps/*federation_name*/usc/self/account/recover/password/validate Specifies the base URL in the e-mail sent to the user during forgotten password validation. The final URL has a query string appended to it.
Profile Update Form	GET	/sps/*federation_name*/usc/self/profile/update Authentication required Requests the profile update form.
Profile Update Submit	POST	/sps/*federation_name*/usc/self/profile/update Authentication required Submits the profile update form.
Account Delete Form	GET	/sps/*federation_name*/usc/self/account/delete Authentication required Requests the account delete form.
Account Delete Submit	POST	/sps/*federation_name*/usc/self/account/delete Authentication required Submits the account delete form.

User Self Care HTTP responses

This topic lists the set of pages that are presented by User Self Care to the user. This set fits into the following categories:

Info
Informational page presenting instructions, errors, or a success statement.

Form
An HTML form for the user to supply data.

Redirect
An HTTP redirect.

Table 124. HTTP Responses

Name	Type	Description
Enrollment Request Form	Form	Gathers the following information: • Requested user ID • E-mail address • Password • Password confirmation • Profile attributes • Captcha input (optional)
Enrollment Validation	Form	Informs the user that an e-mail has been sent for validation purposes or that an error has occurred.
Enrollment Result	Info	Informs the user that their account has been created or that an error has occurred.
Change Password	Form	Gathers the following information: • Old password • New password • New password confirmation
Change Password Result	Info	Informs the user that their password has been changed or that an error has occurred.
Forgotten ID	Form	Gathers the following to help a user recover a forgotten user ID: • E-mail address • Captcha input. This value is optional.
Forgotten Password	Form	Gathers the following information: • User ID • Captcha input This value is optional.
Forgotten Password Secret Question	Form	Shows the secret question. Gathers the following information: • Answer to secret question • New password • New password confirmation • Captcha input This value is optional.
Post Forgotten ID	Info	Presents an error or success statement following attempted recovery of a forgotten ID.
Profile Update	Form	Presents the user with their current profile details and gathers modifications to the fields.

Table 124. HTTP Responses (continued)

Name	Type	Description
Post Profile Management	Info	Presents an error or success statement following profile management operations.
Account Delete	Form	Presents an icon for the user to click to delete their account.
Post Account Delete	Info	Presents an error or success statement following account deletion

Validating form contents

Consider providing client-side input validation to verify that form fields contain data appropriate for their intended type. The provided User Self Care HTML pages contain several examples.

Captcha demonstration

The Captcha demonstration STS Module provides an example of how to integrate Captcha with User Self Care.

User Self Care provides HTML pages that support the user self care operations. Several of these pages are good candidates for the type of input validation that Captcha provides. You can configure these pages to include a macro for Captcha. The User Self Care application can replace the macro value with the HTML source necessary to support the Captcha demonstration. When Captcha is not configured, the macro is not substituted and the Captcha elements are not shown on the page.

When a user initially requests a page that contains a Captcha challenge, the Captcha STS module is contacted. The module randomly selects an image from the set of configured images. This image constructs the macro on the HTML page that is shown to the user.

After the macro substitution occurs, a block of code like the example code in the figure is shown on the page.

```
<label for="demo_captcha">
    Please enter the verification word(s) shown below (required)
</label>
<br />
<img src="http://myserver/public/captcha_test/hello.jpg" border="0" />
<br />
<input type="hidden"
       name="usc.demo.captcha.challenge.field"
       id="usc.demo.captcha.challenge.field"
       value="http://myserver/public/captcha_test/hello.jpg" />
<input style="background-color:#F8F8C8;"
       type="text"
       name="usc.demo.captcha.response.field"
       id="usc.demo.captcha.response.field" />
```

Figure 52. Captcha example

This block provides a `src` tag and two input fields. The `src` tag shows the image to the user. The first input field provides the name of the image. The second gathers the user input, which is the text in the image.

When the form is submitted, the two input fields are provided to the demonstration Captcha STS module. This module compares the user answer with the string that is associated with that image. If a match is correct, the validation is finished.

Note: The first input field specifies a value that is the URL of a server hosting the images that are shown to the user.

The Captcha demonstration package is in the directory:

`Federated_Identity_Manager_installation_directory/examples/demo/captcha`

This directory contains:

- A readme file
- A `com.tivoli.am.fim.demo.sts.captcha.jar` file containing both the compiled code and the source code for the Captcha STS demonstration module.
- A `captchaTestImages` directory containing:
  - A set of six JPEG images
  - A `DemoCaptchaImagesInfo.txt` file that shows the mapping between the image file names and the text string that the user must enter when presented with the associated image.

For configuration instructions, see “Configuring the Captcha demonstration” on page 424.

Chapter 36. Deploying User Self Care

Tivoli Federated Identity Manager automatically installs User Self Care as part of the run time. You are not required to install any additional software, unless you plan to use Tivoli Access Manager as the target user registry.

Administrations who want to deploy User Self Care must be familiar with the administration of:

- WebSphere Application Server, including the **wsadmin** administration interface.
- Tivoli Federated Identity Manager secure token service (STS) modules and trust chains.
- Tivoli Directory Server LDAP.

Administrations who want to use Tivoli Access Manager as the target user registry or WebSEAL as the point of contact serve must be familiar with Tivoli Access Manager for e-business administration.

The following list summarizes the tasks for deploying User Self Care and the order in which to perform them. Before you start a task, ensure that you have finished any prerequisite tasks.

1. Configure a Tivoli Federated Identity Manager domain. The configuration steps include configuring the runtime management.

   The steps for this task are identical for all Tivoli Federated Identity Manager scenarios. There are no tasks in this topic that are unique to User Self Care. The task links point you to common task topics in the *Tivoli Federated Identity Manager Configuration Guide.*

   "Configuring a Tivoli Federated Identity Manager domain" on page 414
2. Integrate User Self Care with the user registry for your deployment. User Self Care supports Tivoli Directory Server and Tivoli Access Manager registries. You are directed to the instructions that match your registry type.

   "Configuring a user registry" on page 414
3. User Self Care configuration relies on values obtained from a response file. In this task, you populate a response file with values applicable to your deployment.

   "Configuring a response file" on page 421
4. You use the response file created in the previous task to configure your User Self Care deployment. This step describes how to view pre-configured trust chains from the administration interface. This step also describes how to use the Tivoli Federated Identity Manager command-line interface to deploy your User Self Care environment. Optionally, you can configure the Captcha demonstration.

   "Configuring User Self Care" on page 423
5. When your deployment includes Tivoli Access Manager WebSEAL server as a point of contact server, you must integrate some User Self Care features with WebSEAL. This set of tasks instructs you how to accomplish the integration.

   "Integrating User Self Care with WebSEAL" on page 427

Configuring a Tivoli Federated Identity Manager domain

You must configure a Tivoli Federated Identity Manager domain.

Before you begin

Install the following Tivoli Federated Identity Manager components:

- Runtime management
- Administration console

Procedure

1. Log in to the administration console.
2. Create a domain. Follow the instructions in Chapter 1, "Domain configuration," on page 3.

What to do next

Continue with "Configuring a user registry"

Configuring a user registry

Integrate User Self Care with the user registry set up for your deployment.

User Self Care supports these registries through WebSphere Federated Repositories configuration:

- IBM Tivoli Directory Server. See "Configuring a Tivoli Directory Server."
- IBM Tivoli Access Manager. See "Configuring a Tivoli Access Manager adapter" on page 415.
- Microsoft Active Directory. See "Configuring an Active Directory server" on page 420.

Configuring a Tivoli Directory Server

Configure WebSphere Federated Repository for Tivoli Directory Server LDAP.

About this task

Do not use this task if you are using Tivoli Access Manager as a user registry. See "Configuring a Tivoli Access Manager adapter" on page 415.

Procedure

1. Log on to the administrative console.
2. Select the Security tab, and select **Global Security**.
3. Click **Configure**.

   The icon is located to the right of the Federated Repositories menu.
4. Click **Add Base Entry to Realm**.
5. Click **Add Repository**.
6. Enter a name for **Repository Identifier**

   You can specify an identifier name.
7. Enter values in the following fields:
   - **Directory Type**
   - **Primary Host Name**

- **Port**
- **Bind distinguished name**
- **Bind password**

You can optionally provide values for additional fields.

8. Click **OK** and save. You now see a page that requests **Distinguished name of a base entry that uniquely identifies this set of entries in the realm**.
9. Enter a base entry name. Click **OK** and save.

   If necessary, see the WebSphere Application Server documentation on WebSphere Federated Repository.

   Note: Remember the base entry name. You must use it when configuring user self care.
   The configuration page for **defaultWIMFileBasedRealm** is shown.
10. Examine the table labeled **Repositories in the realm**. Verify that you new realm is shown, and that the **Base Entry** is set to the value you entered. Click **OK** and save. The administrative console returns to the **Global Security** page.
11. Click the **Enable Application Security** check box.
12. Click **OK** and save.

What to do next

Continue with "Configuring a response file" on page 421.

Configuring a Tivoli Access Manager adapter for WebSphere Federated Repository

To configure a Tivoli Access Manager adapter for User Self Care, you must configure the adapter and then add it to WebSphere Federated Repository as a custom registry.

Complete the following tasks:

1. "Configuring a Tivoli Access Manager adapter."
2. "Configuring the adapter as a WebSphere Application Server custom registry" on page 417.

If necessary, consult the troubleshooting information in "Troubleshooting WebSphere Application Server login failures" on page 419.

Configuring a Tivoli Access Manager adapter

Configure this adapter when User Self Care manages the Tivoli Access Manager registry.

About this task

This adapter uses the Tivoli Access Manager Registry Direct Java API to perform administration commands such as creating users and groups. The Tivoli Access Manager installation provides this adapter.

Note: If you are not using a Tivoli Access Manager adapter, do not use these instructions. See "Configuring a Tivoli Directory Server" on page 414.

Procedure

1. Ensure that you have installed Tivoli Access Manager.

2. Ensure that you have installed and configured Tivoli Access Manager using Tivoli Directory Server as the user registry.
3. Ensure that you have installed the Tivoli Access Manager 6.1.1 Java run time component.
4. Copy *TAM_installation_directory*/java/export/rgy/com.tivoli.pd.rgy.jar to *WebSphere_installation_directory*/lib.
5. Create a Tivoli Access Manager user identity that runs the Java API.

   For example:

   ```
   pdadmin -a sec_master -p sec_master_password
   pdadmin sec_master> user create -no-password-policy user_name
   cn=user_name,registry_suffix user_name user_name password
   ( SecurityGroup ivacld-servers remote-acl-users )
   pdadmin sec_master> user modify user_name account-valid yes
   ```

   In the example, *user_name* is your choice of name for the user. A good naming scheme would be:

   ```
   tamVMMAdapter-machine_name
   ```

   The value *registry_suffix* is the suffix of the registry where this user must be stored. For example:

   ```
   o=ibm,c=us
   ```

6. Go to the computer where the Tivoli Access Manager adapter is to be configured. Change directory to *WebSphere_installation_directory*/lib . Run the **com.tivoli.pd.rgy.until.RgyConfig** tool.

   Use the IBM Java runtime environment to run this tool. For example:

   ```
   <WebSphere install>/AppServer/java/jre/bin/java
   ```

Table 125. Using the com.tivoli.pd.rgy.util.RgyConfig utility

Syntax:

```
java com.tivoli.pd.rgy.util.RgyConfig properties_file_destination create Default
Default "ldaphostname:389:readwrite:5" "DN" DN_password
```

properties_file_destination
Specifies the full path to an existing directory and the name of a file that is created when this command is run. Place the file in a directory appropriate for your WebSphere Application Server deployment:

- For a non-clustered WebSphere Application Server server:

  ```
  WebSphere_application_server/profiles/server_name/config/itfim
  ```

- For a WebSphere Application Server cluster (replicated) environment, create the file on the DMgr:

  ```
  WebSphere_application_server/profiles/DMgr_server_name/config/itfim
  ```

ldaphostname
The host name of the LDAP server to which Tivoli Access Manager is configured. The host name is specified in the Tivoli Access Manager runtime configuration file:

```
Tivoli Access Manager_installation_directory/etc/ldap.conf
```

389
The default LDAP port. Modify as needed for your deployment.

"DN"
The Distinguished Name (DN) specified in the **pdadmin** user creation command. Ensure that the value is surrounded by double quotation marks.

DN_password
The password for the DN.

Example command:

```
java com.tivoli.pd.rgy.util.RgyConfig
WebSphere_application/profiles/<server>/config/itfim/tamVMMAdapter.properties
 create Default Default "myldapsystem:389:readwrite:5"
"cn=tamVMMAdapter-myhost,o=ibm,c=us" mypasswordmypassword
```

7. Update the configuration as needed for your WebSphere Application Server deployment:
   - For a non-clustered WebSphere Application Server server, reload the Tivoli Federated Identity Manager configuration.
   - For a WebSphere Application Server cluster (replicated) environment, perform a full WebSphere Application Server resynchronization, and reload the Tivoli Federated Identity Manager configuration.

What to do next

Continue with “Configuring the adapter as a WebSphere Application Server custom registry.”

Configuring the adapter as a WebSphere Application Server custom registry

To accomplish integration with WebSphere, configure the Tivoli Access Manager adapter as a WebSphere Application Server custom registry.

Before you begin

Complete the task “Configuring a Tivoli Access Manager adapter” on page 415.

About this task

After configuring the Tivoli Access Manager adapter with the Tivoli Access Manager runtime environment, you must configure the Virtual Member Manager (VMM) Tivoli Access Manager Adapter into WebSphere Application Server as a custom registry.

Note: For information about configuring WebSphere Federated Repository custom registries, see the WebSphere Application Server documentation. For WebSphere Application Server Network Deployment 6.1, see the WebSphere information center.

Procedure

1. Stop the WebSphere Application Server.
2. Change directory to:

   ```
   WebSphere_Installation_directory/profiles/profile_name/config/
   cells/cell_name/wim/config
   ```

3. Use a text editor to open `wimconfig.xml`.

   Note: Back up `wimconfig.xml` before you change it.
4. Add a new `config:repositories` element to the file. Place this element before the `config:realmConfiguration` element.

   This entry specifies the class name of the adapter, and sets an identifier for the repository. For example, to specify a class name of `com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter` and to set the `TAMRegistryAdapter` repository as the identifier:

   ```
   <config:repositories
   adapterClassName="com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter"
   id="TAMRegistryAdapter"/>
   ```

5. Save the wimconfig.xml file, and close the text editor.
6. Copy the *`TAM_installation_directory`*`/java/export/vmm_tam_adapter/VMMTamAdapter.jar` file to *`WebSphere_install_directory`*`/lib`.
7. Start **wsadmin** in **no connection** mode:

   ```
   wsadmin -conntype none
   ```

8. Disable paging in the common repository configuration. Set the `supportPaging` parameter for the `updateIdMgrRepository` command to `false` to disable paging.

   ```
   $AdminTask updateIdMgrRepository {-id TAMRegistryAdapter
    -supportPaging false }
   ```

   Note: A warning is shown until the configuration of the sample repository is finished.
9. Add a custom property for the `TAMRegistryAdapter`.

   ```
   $AdminTask setIdMgrCustomProperty {-id TAMRegistryAdapter
   -name tamConfFile -value "properties_file_destination"}
   ```

   properties_file_destination

   The properties file that was created as the result of running **com.tivoli.pd.rgy.util.RgyConfig** in the prerequisite task “Configuring a Tivoli Access Manager adapter” on page 415.
10. Add a base entry to the adapter configuration using the **addIdMgrRepositoryBaseEntry** command to specify the name of the base entry for the specified repository:

```
$AdminTask addIdMgrRepositoryBaseEntry {-id TAMRegistryAdapter
-name base_entry_name }
```

base_entry_name
   This name must match the suffix used by the Tivoli Access Manager user registry.

11. Use the **addIdMgrRealmBaseEntry** command to add the base entry to the realm. This action links the realm with the repository.

    ```
    $AdminTask addIdMgrRealmBaseEntry {-name defaultWIMFileBasedRealm
    -baseEntry base_entry_name }
    ```

    base_entry_name
       This name must match the value you specified in the previous command.

    defaultWIMFileBasedRealm
       The default realm name is `defaultWIMFileBasedRealm`. If this realm name was renamed, use the new realm name instead of `defaultWIMFileBasedRealm`.

12. Save your configuration changes. Enter the following commands to save the new configuration and close the **wsadmin** tool:

    ```
    $AdminConfig save
    exit
    ```

13. Restart the WebSphere Application Server.

What to do next

Select one:

- If you can successfully log on to WebSphere Application Server, continue with “Configuring a response file” on page 421
- If you cannot log on to WebSphere Application Server, see “Troubleshooting WebSphere Application Server login failures”

Troubleshooting WebSphere Application Server login failures

If you cannot log on to WebSphere Application Server following configuration of the adapter, review these troubleshooting tips.

About this task

If a registry cannot be contacted, WebSphere Application Server prevents you from logging on. This limitation occurs even if the WebSphere Application Server administration account is located in a different registry. Misconfiguration or lack of availability of a required registry can result in WebSphere Application Server preventing you from logging in as the administrator.

If you encounter this problem after configuring the Tivoli Access Manager adapter, try the following troubleshooting steps:

Procedure

1. Ensure that the Tivoli Access Manager registry is available. Since Tivoli Access Manager Registry adapter does not maintain an authentication cache, you see a "cannot log in" error immediately when the registry is unavailable.
   a. Use **pdadmin** to connect to the registry and perform a test user creation to confirm.
   b. Restart the registry and correct any connection issues if necessary.
   c. If the problem persists, continue to the next step.

2. Open the `wimconfig.xml` file and verify the settings in the new code that you created.

```
<config:repositories adapterClassName="com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter"
id="TAMRegistryAdapter" supportPaging="false">
<config:baseEntries name="o=ibm,c=us"/>
<config:CustomProperties
name="tamConfFile"
value="/opt/IBM/WebSphere/AppServer/profiles/dmgr/config/itfim/tamVMMAdapter.properties"/>
</config:repositories>
```

Figure 53. Sample wimconfig.xml settings

   - Confirm that the location or name of the properties file is correct.
   - Confirm that the suffix is correct for the Tivoli Access Manager registry.

   Note: If you modify the configuration file, you must restart WebSphere Application Server. WebSphere Application Server requires you to log in as the administrator to stop WebSphere Application Server. However, if you cannot log in you must stop the WebSphere Application Server process. You can then restart WebSphere Application Server without a login.

3. If in the previous step you did not identify any problems with the configuration file, roll back to the backup copy of wimconfig.xml.
   a. Make a backup of your new wimconfig.xml file.
   b. Restore the backup of the original wimconfig.xml file.
   c. Restart WebSphere Application Server.

      Note: WebSphere Application Server requires you to log in as the administrator to stop WebSphere Application Server. However, if you cannot log in you must stop the WebSphere Application Server process. You can then restart WebSphere Application Server without a login.

   If you can log in after restoring the backed up file, there is a problem with the Tivoli Access Manager adapter configuration. Review the configuration and correct any errors.

Configuring an Active Directory server

Configure WebSphere Federated Repository for Microsoft Active Directory.

About this task

Do not use this task if you are using Tivoli Access Manager as a user registry. See “Configuring a Tivoli Access Manager adapter” on page 415.

Procedure

1. Log on to the administrative console.
2. Select the Security tab, and select Global Security.
3. Click **Configure**.

   The icon is located to the right of the Federated Repositories menu.
4. Click **Add Base Entry to Realm**.
5. Click **Add Repository**.
6. Enter a name for **Repository Identifier**

   You can specify an identifier name.

7. Enter values in the following fields:
   - **Directory Type**
   - **Primary Host Name**
   - **Port**
   - **Bind distinguished name**
   - **Bind password**

   You can optionally provide values for additional fields.
8. On the WebSphere Application Server console, select **Require SSL communications**.

   Note: Configuration of SSL communication between a WebSphere Application Server and a user registry such as Active Directory requires additional steps. See the documentation for your version of WebSphere Application Server for instructions on configuring SSL connections with WebSphere Application Server.
9. Click **OK** and save. You now see a page that requests **Distinguished name of a base entry that uniquely identifies this set of entries in the realm**.
10. Enter a base entry name. Click **OK** and save.

    If necessary, see the WebSphere Application Server documentation on WebSphere Federated Repository.

    Note: Remember the base entry name. You must use it when configuring user self care.
    The configuration page for **defaultWIMFileBasedRealm** is shown.
11. Examine the table labeled **Repositories in the realm**. Verify that you new realm is shown, and that the **Base Entry** is set to the value you entered. Click **OK** and save. The administrative console returns to the **Global Security** page.
12. Click the **Enable Application Security** check box.
13. Click **OK** and save.

What to do next

Continue with "Configuring a response file."

Configuring a response file

Create a response file and then populate it with values specific to your deployment.

About this task

User Self Care loads configuration from an XML properties file called a *response file*. This file contains the responses to configuration options. In most cases, response file contents are generated by administrator choices during initial deployment. For user self care, loading a properties file is required as part of initial configuration.

Procedure

1. Create a response file as needed for your deployment. Use **wsadmin**:

   ```
   $AdminTask manageItfimUserSelfCare {-operation createResponseFile
   -fileId target_location }
   ```

   The value *target_location* is fully qualified path to a file that is created. The value *target_location* is fully qualified path to a file that is created.

2. Determine a value of each parameter in the response file, as required by your deployment.

   Optionally, use the following worksheet to plan your response file. The worksheet identifies the required parameters. In the response file, you can search for the string `REQUIRED` to find these parameters.

   For more information about each parameter in this worksheet, see Chapter 38, "Response file parameters," on page 437

Table 126. User Self Care response file parameters

Response file parameter	Required?	Default Value	Your value
AccountCreateLifetime	yes	86400	
AccountRecoveryFailureLifetime	no	86400	
AccountRecoveryFailureLimit	no	3	
AccountRecoveryFailureLockoutTime	no	86400	
AccountRecoveryLookupAttribute	no	mail	
AccountRecoveryLookupField	no	none	This field is deprecated.
AccountRecoveryValidationAttributes	no	mail	
AccountRecoveryValidationLifetime	no	86400	
AttributeMappingFilename	yes	none	
BaseURL	yes	none	
CaptchaSTSModuleId	yes	default-usc-captcha-noop	
DemoCaptchaImageAndKeyList	Yes, if using Captcha	Fixed content.	Do not modify.
DemoCaptchaImageRootURL	Yes if using Captcha	none	
EnrollmentEmailSender	yes	none	
EntitySuffix	yes	o=ibm,c=us	
GroupMembershipGroups	no	none	
PasswordRecoveryEmailSender	yes	none	
ProfileManagementAttributes	yes	businessCategory roomNumber mobile mail	
SMTPAuthenticatePassword	no, unless your SMTP server requires it	none	
SMTPAuthenticateUsername	no, unless your SMTP server requires it	none	
SMTPServerName	yes	none	

3. Update your response file with the values and save the file.

What to do next

Continue with the topic: “Configuring User Self Care.”

Configuring User Self Care

Follow the steps in this topic to configure user self care with the Tivoli Federated Identity Manager deployment.

Before you begin

Ensure that you have finished the prerequisite configuration tasks:

1. “Configuring a Tivoli Federated Identity Manager domain” on page 414
2. “Configuring a user registry” on page 414
3. “Configuring a response file” on page 421

About this task

Do the following tasks in order. The instructions for each task provide a link to the next task. The list of tasks is shown here as an overview.

Procedure

1. “Showing trust chains”
2. “Configuring the Captcha demonstration” on page 424

   Skip this step if you do not intend to use the Captcha demonstration.
3. “Using a response file to configure User Self Care” on page 425
4. “Configuring a point of contact server” on page 425
5. “Integrating User Self Care with WebSEAL” on page 427

   Skip this step if you are using WebSphere Application Server as the point of contact server

Showing trust chains

You can configure Tivoli Federated Identity Manager to show the trust chains that are created by default for User Self Care.

About this task

You can use the administrative console to view each trust chain. Trust chains correspond to one or more User Self Care operations. When you view the trust chains, you see the STS modules that accomplish the operation. You can then customize the modules and chains to fit your deployment.

Note: For information about how to customize User Self Care, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis/display/tivolifederatedidentitymanager/Home

Procedure

1. Log on to the administrative console.
2. Go the Runtime Node Management panel.

3. In the custom property part of the panel, select the menu entry for `STS.showUSCChains`.
4. Set the value to `true`.
5. Save the configuration.
6. When prompted, load the configuration changes.
7. Restart WebSphere Application Server to refresh the management commands available to **wsadmin**.

What to do next

Select one of the following steps:

- If you want to use the Captcha demonstration, continue with "Configuring the Captcha demonstration."
- If you do not want to use the Captcha demonstration, continue with "Using a response file to configure User Self Care" on page 425.

Configuring the Captcha demonstration

You can optionally configure the Captcha demonstration as part of your User Self Care deployment.

Before you begin

Ensure that you have finished all of the prerequisite configuration steps:

- "Configuring a Tivoli Federated Identity Manager domain" on page 414
- "Configuring a user registry" on page 414
- "Configuring a response file" on page 421
- "Showing trust chains" on page 423

Procedure

1. Host the provided image files on a web server that is accessible to your users.

   Ensure that you know the location of the root URL of the images that are used during the configuration of the Captcha STS module. The value is stored in the DemoCaptchaImageRootURL parameter in the response file.
2. Activate the plug-in:
   a. Copy the Captcha jar file to the Tivoli Federated Identity Manager plug-ins directory. For example, copy:

      ```
      FIM_install_dir/examples/demo/
      captcha/com.tivoli.am.fim.demo.sts.captcha.jar
      ```

      to the directory:

      ```
      TFIM_install_dir/plugins
      ```

   b. Using the Runtime Node Management Panel, click the **Publish Plugins** icon.
   c. Click **Load Configuration Changes**.
3. Use the Module Instances Panel to create an instance of the DemoCaptchaSTSModule. Set the Module Instance Name to the value `usc-captcha-demo`.

What to do next

Continue with "Using a response file to configure User Self Care" on page 425.

Using a response file to configure User Self Care

Use the response file that you created previously to supply the necessary properties to the configuration command for user self care.

Before you begin

Ensure that you have finished the prerequisite configuration tasks:

- "Configuring a Tivoli Federated Identity Manager domain" on page 414
- "Configuring a user registry" on page 414
- "Configuring a response file" on page 421
- "Showing trust chains" on page 423
- If you are using the Captcha demonstration, ensure that it is configured. See "Configuring the Captcha demonstration" on page 424

Procedure

1. Obtain your configured response file.
2. Run **wsadmin**

   ```
   wsadmin.sh -username WebSphere_adminstrator_name -password password
   ```

3. Load the response file:

   ```
   $AdminTask manageItfimUserSelfCare {-operation configure -fimDomainName
   domain_name -federationName federation_name
    -fileId response_file_path }
   ```

   Supply these values:

   domain_name
   The name of the Tivoli Federated Identity Manager domain that you created.

   federation_name
   The name of the Tivoli Federated Identity Manager federation that you created.

   response_file_path
   The location of your User Self Care response file.

4. Reload the Tivoli Federated Identity Manager configuration.

   ```
   $AdminTask reloadItfimRuntime {-fimDomainName domain_name }
   ```

   Supply this value:

   domain_name
   The name of the Tivoli Federated Identity Manager domain that you created.

What to do next

Continue with the topic: "Configuring a point of contact server."

Configuring a point of contact server

You must configure a point of contact server for User Self Care.

Select the instructions for the type of point of contact server that your deployment uses:

- "Configuring WebSphere Application Server as a point of contact server" on page 426
- "Configuring WebSEAL as a point of contact server" on page 426

Configuring WebSphere Application Server as a point of contact server

You can configure WebSphere Application Server as the point of contact server for User Self Care.

Procedure

1. Use **wsadmin** to activate the **WebSphere** point of contact type.

   Use the **wsadmin** commands:

   ```
   $AdminTask manageItfimPointOfContact {-operation activate
   -uuid uuid4f3d17d-0106-w412-r36b-a0d5ecc604ba
   -fimDomainName your_domain_name}
   ```

   ```
   $AdminTask reloadItfimRuntime {-fimDomainName your_domain_name}
   ```

2. Log on to the administrative console.
3. Select **Enterprise Applications > ITFIMRuntime > Security role to user/group mapping**.
4. Update the **FIMUserSelfCareAnyAuthenticated** application role to be **AnyAuthenticated**.
5. Save the WebSphere Application Server configuration.
6. Restart WebSphere Application Server.

What to do next

Review the performance tuning guidelines in Chapter 37, “Tuning User Self Care,” on page 433.

Configuring WebSEAL as a point of contact server

You can configure WebSEAL as the point of contact server for User Self Care.

Procedure

1. Determine the location of your tfimcfg.jar file.

   This file is located in the hierarchy under the Tivoli Federated Identity Manager installation directory. On UNIX, the path is:

   ```
   /opt/IBM/FIM/tools/tamcfg/tfimcfg.jar
   ```

2. Run the **tfimcfg** tool.

   ```
   java -jar tfimcfg.jar -cfgfile /opt/pdweb/etc/webseald-default.conf
   -action tamconfig
   ```

   Usage notes:

   - The file paths might differ for your installation and your WebSEAL instance
   - The default Tivoli Federated Identity Manager HTTP port is 9080. This port is also the WC_defaulthost port for the WebSphere Application Server.
   - Do not specify an optional Tivoli Federated Identity Manager administrator user ID or password
   - Answerno to the question `Use SSL connection to ITFIM server`.
   - Select `uscfed` from the list of Federations to configure.

What to do next

Continue with the topic: “Integrating User Self Care with WebSEAL” on page 427.

Integrating User Self Care with WebSEAL

User Self Care deployments that have a Tivoli Access Manager registry in most cases use WebSEAL as a point of contact server. In this scenario, you must integrate interactions between two components that accomplish the task of account deletion and password management.

- Account deletion

  When a user deletes their account from the Tivoli Access Manager registry, as part of a user self deployment, ensure that their current session is ended. This restriction is required for best security practice

  Deletion of the user session includes their WebSEAL session. User Self Care by default terminates the WebSEAL session when the account is deleted. However, this termination is dependent on your prior use of the tfimcfg tool to configure WebSEAL as point of contact server. If you have run this tool as instructed earlier, no special configuration is required.

  If you have not configured WebSEAL as a point of contact server, do so now. See "Configuring WebSEAL as a point of contact server" on page 426
- Password management

  Two password management operations are affected when WebSEAL is the point of contact server. The operations are: Change Password and Expired Password. Both of these integrations require that you permit unauthenticated access to the change password page and use a modified User Self Care Change Password form.

Integrating the change password operation with WebSEAL

When WebSEAL is the point of contact server and a user wants to change a password, the user must provide data. There are several ways that the user can do this task.

Two ways to provide the data are:

- The user can directly access the User Self Care Change Password URL
- The WebSEAL change password form can redirect the user to the User Self Care Change Password form. You can add a meta tag redirect in the WebSEAL change password page to support this action.

Integrating the expired password operation with WebSEAL

WebSEAL as a point of contact server manages authentication, including expired passwords. However, when User Self Care is integrated with WebSEAL, it must manage the handling of expired passwords.

When this scenario is the case, the following steps occur:

1. WebSEAL flags the authenticated session as `expired`.
2. The user is presented with a modified version of the WebSEAL expired password form
3. The user provides input and submits the expired password form. This action POSTs the password data to the User Self Care change password URI.

   Note: The password data can be presented by anything. It must meet certain criteria and POST to the correct User Self Care target URL. When the user has submitted the form, User Self Care processes the form contents and handles any errors. This processing can include showing the User Self Care change password form to the user with error details.

4. User Self Care handles the changing of the password.
5. The WebSEAL session is terminated.

   Note: The WebSEAL session is terminated because the session entry managed by WebSEAL is flagged as `expired`. Until this flag is changed, the user is always presented with the WebSEAL change password form. The user will be unable to continue, even after having changed their password using User Self Care. Terminating the session is also a preferred security practice because it requires the user to log in with their new password, in order to continue.
6. The user is shown the User Self Care password change success page. This page can be modified to redirect back to WebSEAL if wanted.

Configuration steps

Do each of the following steps for the operation you want to integrate with WebSEAL:

- To integrate the change password operation with WebSEAL:
  1. "Permitting unauthenticated access to the User Self Care change password form"
  2. "Modifying the user self care WebSEAL change password form"
- To integrate the expired password operation with WebSEAL:
  1. "Permitting unauthenticated access to the User Self Care change password form"
  2. "Modifying the user self care WebSEAL change password form"
  3. "Modifying a WebSEAL expired password form" on page 429
  4. "Supporting redirection back to WebSEAL" on page 430

Permitting unauthenticated access to the User Self Care change password form

To support WebSEAL password operation integration, unauthenticated users must be able to access the Change Password URI through a WebSEAL junction. The junction must be configured with SSL for privacy and confidentiality.

Use pdadmin to permit unauthenticated access to the User Self Care change password form located at:

```
WebSEAL_server/fim_junction/sps/uscfed/usc/self/password/update
```

Consult the Tivoli Access Manager documentation for information about the **pdadmin** command.

When you change this access, you must use a modified User Self Care Change Password form. Continue with "Modifying the user self care WebSEAL change password form"

Modifying the user self care WebSEAL change password form

The user ID must be supplied in the Change Password form when integrating User Self Care change password operations with WebSEAL.

About this task

Permitting unauthenticated access means that it is possible for users to access the change password form. From a security perspective, this user action is acceptable because the user must enter their old password on this form before they can change it. However, you must modify to the default User Self Care form to activate this function.

By default, User Self Care does not require users to enter their user ID on the change password form. Instead, User Self Care gathers it from the authenticated context. This mechanism does not work if the user does not authenticate before requesting the form. If the user requests the form without authenticating, User Self Care returns an error message indicating that no authenticated user identity is available.

To avoid this error, the user ID must be supplied in the Change Password form when integrating User Self Care change password operations with WebSEAL.

Procedure

1. Make a backup copy of `*FIM_install_dir*/pages/C/usc/password/changepassword.html`
2. Copy the example file changepassword.html to the User Self Care pages repository.
   - The example file is:

     FIM_install_dir`/examples/examples/html/usc/password/changepassword.html`
   - The destination location is:

     FIM_install_dir`/pages/C/usc/password/changepassword.html`
3. Log on to the administrative console.
4. Go to the **Runtime Node Management** panel. Click **Refresh Pages**.
5. Save the configuration changes.

What to do next

- If you are integrating the change password operation, you have finished the task.
- If you are integrating the expired password operation, Continue with "Modifying a WebSEAL expired password form"

Modifying a WebSEAL expired password form

Modify the WebSEAL expired password form to ensure correct handling of passwords with User Self Care.

Before you begin

Ensure that you have finished the prerequisite tasks:

1. "Permitting unauthenticated access to the User Self Care change password form" on page 428
2. "Modifying the user self care WebSEAL change password form" on page 428

About this task

There is more than one way to modify the form. The following method is easy.

Procedure

1. Copy the User Self Care changepassword.html file to the WebSEAL directory where the management pages are located. Rename it to usc_changepassword.html.

   For example:

   ```
   /opt/pdweb/www-default/lib/html/C/usc_changepassword.html
   ```

2. Edit the usc_changepassword.html form as follows:
   a. Add a new hidden field:

      ```
      <input type="hidden" name="usc.form.password.expired.flag" value="true" />
      ```

   b. Add another new hidden field:

      ```
      <input type="hidden" name="usc.form.userid" value="%USERNAME%" />
      ```

   c. Remove or comment out the two lines:

      ```
      <div class="hidden" id="errorDiv"> </div>
      <div class="hidden" id="errorAttrDiv"> </div>
      ```

   d. Replace the form ACTION macro with the URL of the User Self Care change password target.

      For example:

      ```
      https://webseal.example.com/fimjct/sps/uscfed/usc/self/password/update
      ```

3. Set the file permissions and ownership of usc_changepassword.html to match the permissions of the other WebSEAL management files.
4. Edit the WebSEAL configuration file. Go to the acnt-mgt stanza and change passwd-expired = passwd_exp.html to passwd-expired = usc_changepassword.html
5. Restart WebSEAL.

What to do next

Optionally, continue with "Supporting redirection back to WebSEAL"

Supporting redirection back to WebSEAL

Optionally, you can direct users back to WebSEAL after they have changed their password.

About this task

In some cases, you might want to host a *landing page* with links to destinations from the WebSEAL system rather than the User Self Care system.

Procedure

1. Create a *password change success* page in the WebSEAL docs directory.

   This page is the landing page at WebSEAL. It can say: Your password has been successfully changed, you will have to login again to access any protected pages.
2. Modify the User Self Care page located at *FIM_install_dir*/pages/C/usc/password/changepassword_success.html to add a meta-redirect tag that redirects the client to the new WebSEAL *password change success* page.

Modifying a User Self Care federation

There are some limitations on how you can modify existing User Self Care federations.

- The command line interface does not support modification of User Self Care federations. Use the administration console to set the runtime property STS.showUSCChains to `true`. View the User Self Care trust chains and modify the trust chains and properties as needed.

  Note that as an alternative you can configure User Self Care by repeating the initial deployment steps. In this case, you must create and edit a new response file, and then use the command line interface to deploy the federation.
- You cannot capture, within a response file, configuration settings that are specific to a particular chain. For example, Attribute Mapping STS modules use a mapping rule file. Different chains might have different mapping rules. You cannot specify the different mapping rules when creating a response file from an existing configuration.

  Parameters that can be specific to a particular chain do not have values set in the response file. When different chains have different mapping rules, use the administration console to modify the chain modules to use different rules files

Unconfiguring User Self Care

Use **wsadmin** to unconfigure User Self Care.

About this task

This task deletes the User Self Care trust chains and the User Self Care federation.

Procedure

1. Start **wsadmin**.
2. Issue the command:

   ```
   $AdminTask manageItfimUserSelfCare {-operation unconfigure
   -fimDomainName your_domain_name -federationName uscfed}
   ```

Chapter 37. Tuning User Self Care

You can improve User Self Care performance by adjusting settings for several distributed caches.

User Self Care maintains three different distributed caches:

- Account Create Cache
- Forgotten Password Cache
- Secret Question Failure Cache

The caches are shared among WebSphere Application Server cluster members to permit a user operation to be properly handled. This sharing is required in case different phases of the operation take place on different nodes.

User Self Care uses the WebSphere Distributed Object Cache technology for implementation of the caches. See the WebSphere Application Server documentation for details on this caching technology.

There are two parameter types that affect each User Self Care distributed cache:

Entry lifetimes
These parameters are set in the User Self Care response file. Cache entries are retained until either the lifetime is hit or the user finishes the operation requiring the cache entry. The names and settings of these cache-specific parameters are described in the individual cache tuning descriptions later in this set of topics.

Cache sizes
These parameters are set in the administrative console by accessing **Resources > Cache Instances > Object Cache Instances**. The `Cache size` parameter controls how many concurrent entries are retained in the cache. The names and settings of these cache-specific parameters are described in the individual cache tuning descriptions

You must size the caches adequately so users can perform operations that require a distributed cache in the configured time period. If a cache is too small, users might not be able to validate their accounts or recover their passwords during the specified time period. You can specify the time period in the configuration for lifetime of the cache entries.

For example, to give your users two minutes to finish an account recovery validation, configure the entry lifetime for the account recovery validation cache to be two minutes. If you expect two users per second to perform an account recovery operation, set the account recovery validation cache to at least 240.

Determine the appropriate size using the following calculation:

```
120 seconds x 2 users/second = 240
```

The default size of the account recovery validation cache is 1000 entries. This default would be adequate for this example. Other operations, such as account creation, might require an increase in the cache size.

Depending on your expected system usage, you might increase the size of one or more caches. This adjustment can affect your hardware requirements. Cache entries take up memory and must be replicated between systems in the cluster.

A preferred performance tuning practice is to provide a buffer for the expected cache size.

See the following topics:

- "Account create cache"
- "Forgotten password cache"
- "Secret question failure cache" on page 435
- "Notes about tuning caches" on page 435

Account create cache

This cache stores the data from the user inputs during the account creation and e-mail validation process. When the user finishes the validation, the User Self Care recovers the data from the cache to create an account in the registry.

Table 127. Account create cache parameters

Parameter	Description
AccountCreateLifetime	Entry lifetimes are controlled by the `AccountCreateLifetime` parameter described in the topic:Chapter 38, "Response file parameters," on page 437
itfim-usc_accountcreate	Cache size is controlled by the `itfim-usc_accountcreate` cache size.

Unlike other operations, each account creation operation creates two cache entries. One entry is consists only of the user ID and a key. The second entry consists of all the data that the user enters in the account creation form.

You configure cache entry lifetimes to be 120 seconds. You expect a peak number of users enrolling during a new application provisioning operation to be 10 each/second. You might want to size your cache as follows:

```
10 users/second x 2 entries/user x 120 seconds/entry = 2400 x 20% buffer ~= 3000.
```

Forgotten password cache

This cache stores the user ID during the Forgotten Password validation operation.

Table 128. Forgotten password cache parameters

Parameter	Description
AccountRecoveryValidationLifetime	Entry lifetimes are controlled by the `AccountRecoveryValidationLifetime` parameter described in the topic: Chapter 38, "Response file parameters," on page 437.
itfim-usc_forgottenpassword	Cache size is controlled by the `itfim-usc_forgottenpassword` cache size. This entry is small, consisting essentially of the user ID and a key.

Secret question failure cache

This cache stores the number of failed secret question answer attempts that have been performed.

Table 129. Secret question failure cache parameters

Parameter	Description
AccountRecoveryFailureLifetime	Entry lifetimes are controlled by the `AccountRecoveryFailureLifetime` parameter described in the topic: Chapter 38, "Response file parameters," on page 437
itfim-usc_secretquestionfailures	Cache size is controlled by the `itfim-usc_secretquestionfailures` cache size. This entry is consists only of a number and a key.

Notes about tuning caches

Configuration of WebSphere Application Server operations can improve your tuning of the caches.

- Replication

  WebSphere does not automatically replicate all of the cached data between nodes. Instead, it just replicates the keys between nodes and only retrieves the data when requested by a particular node. If a key is requested on a particular node system that is not found in the cache, User Self Care attempts the cache lookup operation. The attempt provides time for WebSphere Application Server to finish any possible replication.
- Cache flushes

  Restarting WebSphere Application Server clears caches and returns them to a clean state.
- Removal of User Self Care caches

  Cache entries are retained until either the entry lifetime is hit or the user finishes the operation requiring the cache entry.

Chapter 38. Response file parameters

Use the parameters described in this section to configure response files for User Self Care.

AccountCreateLifetime

Specifies the amount of time, in seconds, that User Self Care recognizes the account creation request as valid, and retain the request in the internal cache. If the Create Account trust chain does not finish account creation in the specified time, the request is discarded and account creation terminates.

This property is required.
Type: Integer
Default: 86400
Maximum: none
Minimum: 0

A setting of '0' disables account creations because entries are not retained in the cache. Larger settings can affect memory consumption and potentially affect performance in replicated environments due to increased data being replicated using DynaCache across nodes.

When setting this property, also consider an appropriate size for the `itfim-usc_accountcreate` cache. See: Chapter 37, “Tuning User Self Care,” on page 433.

AccountRecoveryFailureLifetime

Specifies how long, in seconds, the program retains record of an unsuccessful account validation attempt. When the specified time period elapses, the record of the unsuccessful attempt is discarded, and the counter is decremented by one.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0. The value 0 means to disable locking.

When setting this property, also consider an appropriate size for the `itfim-usc_secretquestionfailures` cache. This parameter is configured separately as part of tuning User Self Care. See: Chapter 37, “Tuning User Self Care,” on page 433.

AccountRecoveryFailureLimit

Specifies the number of times a user can attempt but fail to restore account access before the program locks the account. If the user does not supply a correct answer to the secret question, account access is not restored. When the user fails to restore account access, the value of this property increments by one. When the value equals the specified number, the program locks the account.

Type: Integer
Default: 3
Maximum: none
Minimum: 0.

A setting of 0 or 1 for the minimum causes the account to be locked after the first failure.

AccountRecoveryFailureLockoutTime
Specifies how long, in seconds, the program keeps the account locked after the user has exceeded the maximum number of unsuccessful validation attempts. When the program has locked the account, this value specifies the amount of time that must pass before the program unlocks the account.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0. The value 0 disables locking.

AccountRecoveryLookupAttribute
Specifies an attribute that User Self Care uses as the message target during the forgotten password operation. By default, User Self Care sets this attribute to the mail registry attribute. User Self Care sends e-mail to the value found for the specified registry attribute.

Type: string
Default: `mail`

AccountRecoveryLookupField
This field is deprecated. Do not modify.

AccountRecoveryValidationAttributes
Specifies a user attribute used for user ID lookup. This property specifies a single attribute that the user enters in the Forgotten user ID form, in order to retrieve their user ID (identity). User Self Care uses this registry attribute as a lookup field. User Self Care searches for an entry that contains the attribute supplied by the user, and returns the matching user ID.

Type: String
Default: `mail`

The value you enter here is the actual name of the registry attribute. The attribute mapping secure token service module does not map this parameter when it maps end-user supplied input fields to registry attributes. The attribute name `mail` is the standard LDAP field used for the e-mail address.

AccountRecoveryValidationLifetime
Specifies the amount of time, in seconds, that User Self Care considers the account validation request to be valid.

During password recovery, users must finish a validation step before recovering their password. The validation step consists of responding to a user self care e-mail that specifies a link to access. If the user does not respond within the time period specified by this parameter, the program invalidates the link in the e-mail.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0

A setting of 0 for the minimum disables the ability to recover an account.

When setting this property, also consider an appropriate size for the `itfim-usc_forgottenpassword` cache. This parameter is configured separately as part of tuning User Self Care. See: Chapter 37, "Tuning User Self Care," on page 433.

AttributeMappingFilename

Specifies the path to the location of a file that contains the transformation rules for use with the Attribute Mapping STS Module. This file can be either a JavaScript or XSLT file.

User Self Care ships with a default JavaScript file named `usc.js`:

```
Federated_Identity_Manager_installation_dir/examples/js_mappings
```

This property is required.
Type: String
Default: none

Example:

```
/opt/IBM/FIM/examples/js_mappings/usc.js
```

BaseURL

Specifies a fully qualified URL for the root of the User Self Care federation. User Self Care uses the root to construct dynamic HTML elements. The syntax is as follows:

```
method//POC_server:port/FIM_junction/sps
```

Where:

method

Must be either `http:`or `https:`

POC_server:port

The fully qualified host name, and optional port number, of the point of contact server.

FIM_junction

The name of the WebSEAL junction. This value is only required when using a WebSEAL point of contact server.

This property is required.
Type: String
Default: none

Example:

```
https://myWebSEALserver.example.com/myTFIMjct/sps
```

Note: If you are using WebSEAL as a point of contact server, you likely have not yet created a junction to the Tivoli Federated Identity Manager server. In most cases, you create this junction at the end of the User Self Care configuration steps. However, you must determine the name of the junction now, so that you can set the BaseURL value in the response file now. You must remember the junction name, for use later when running the **tfimcfg** command.

CaptchaSTSModuleId

Specifies either the demonstration Captcha module, or specifies a placeholder module that takes no action. When this value is specified, User Self Care activates the demonstration Captcha module.

This property is required.
Type: String
Default: none

There are two valid values for this field:

- `usc-captcha-demo`

  Use this value if you want to activate the demonstration Captcha module. If you use this setting, you must set the other Captcha settings in this response file. To use the Captcha demonstration, you must also configure the module. See: "Configuring the Captcha demonstration" on page 424.
- `default-usc-captcha-noop`

  Use this value if you want to use the placeholder module `USCNoOpsSTSModule`. This module takes no action, but serves as a placeholder for a customer-provided validation module that can be used, as an example, for Captcha validation. The USCNoOpsSTSModule makes it easier for customers to provide their own module without redefining the trust chains.

DemoCaptchaImageAndKeyList

This field is required if you are using the Captcha demonstration module.

The contents are fixed and must not be modified.

Note: The `DemoCaptchaImageAndKeyList` parameter has already been set. The program ignores this parameter if you are not using the demonstration Captcha module.

DemoCaptchaImageRootURL

Specifies a URL of a directory that contains the images used for the demonstration Captcha module provided with User Self Care.

You must specify a value for this property if you want to use the Captcha demonstration module.

Example:

```
https://images.example.com/captcha/demo
```

EnrollmentEmailSender

Specifies a fully qualified e-mail address for the account that User Self Care uses to send a message to the user. The message validates the user enrollment. In most cases, this address is an e-mail address that does not receive responses.

This property is required.
Type: string
Default: none

Example:

```
no-reply@example.com
```

EntitySuffix

Specifies a suffix where created users are stored in the registry. This suffix must uniquely identify the registry that User Self Care uses for all operations.

This property is required.
Type: String
Default: `o=ibm,c=us`

GroupMembershipGroups

Specifies a list of groups to which to add newly enrolled users. Specifies one or more groups that are defined in the user registry used by the Create Account trust chain. The group names are specific to the user registry.

Type: String
Default: none

Example:

```
<void method="add">
 <string>Group1</string>
</void>
<void method="add">
 <string>Group2</string>
</void>
```

PasswordRecoveryEmailSender

Specifies a fully qualified e-mail address for the User Self Care account that sends a message to the user. User Self Care uses the message to validate a password recovery operation. In most cases, this e-mail address does not receive responses.

This property is required.
Type: string
Default: none

Example:

```
no-reply@example.com
```

ProfileManagementAttributes

Defines the set of registry attributes that are used for profile information. In order to provide a working prototype, the user self care solution defines a set of registry attributes for use with the default function. User Self Care does not modify the schema of the target registry. For this reason, the number of profile attributes are limited and use standard LDAP attributes that are present in most cases.

This property is required. The list of attributes used are:

- businessCategory
- roomNumber
- mobile
- mail

The attributes are represented in the configuration file as follows:

```
<object class="java.util.ArrayList">
 <void method="add">
  <string>businessCategory</string>
 </void>
 <void method="add">
  <string>roomNumber</string>
 </void>
 <void method="add">
       <string>mail</string>
     </void>
 <void method="add">
       <string>mobile</string>
     </void>
</object>
```

Figure 54. Profile management attributes in the response file

SMTPAuthenticatePassword

The password for the account specified by the SMTPAuthenticateUsername parameter if using authentication to the SMTP server. This property is optional.

Type: string
Default: none

SMTPAuthenticateUsername

The user name that authenticates to the SMTP server. This property is optional.

Type: string
Default: none

SMTPServerName

The fully qualified host name of the Simple Mail Transport Protocol (SMTP) server that sends e-mail for the user. This property is required.

Type: string
Default: none

Part 6. Customization

The topics in the Customization section explain how to customize components and functions of Tivoli Federated Identity Manager to better suit your environment.

Chapter 39. Customizing runtime properties

Custom properties can be used to tailor the runtime service of the Tivoli Federated Identity Manager to meet specific needs.

The use of custom properties is an advanced task. You might need to be familiar with broader topics about the architecture and services of Tivoli Federated Identity Manager, in order to understand how to use custom properties. Refer to the Tivoli Federated Identity Manager information center for more information: http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp

Creating a custom property

You can customize the domain configuration by defining a custom property.

About this task

The syntax for custom properties is:

```
property_name = property_value
```

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Domain Management** → **Runtime Node Management**.
2. The Runtime Node Management panel is displayed. Click **Runtime Custom Properties**. The Runtime Custom Properties panel is displayed.
3. Select the scope of the custom property, either cell or node, from the **Scope** list. A list of properties at the scope you selected is displayed.
4. Click **Create**. A list item is added to the list of properties with the name of **new key** and a value of **new value**.
5. Select the placeholder property.
6. Enter a string in the **Name** field. Do not insert the space character in this field.
7. Enter a string in the **Value** field. Spaces are allowed in this field.
8. Click **OK** to apply the changes that you have made and exit from the panel.

Deleting a custom property

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Domain Management** → **Runtime Node Management**.
2. The Runtime Node Management panel is displayed. Click **Runtime Custom Properties**. The Runtime Custom Properties panel is displayed.
3. Select the scope of the custom property, either cell or node, from the **Scope** list. A list of properties at the scope you selected is displayed.
4. Select a name and value pair.
5. Click **Delete**. The panel refreshes and the name and value pair is removed from the list of custom properties.
6. Choose one of the following actions:
   - Click **Apply** to apply the changes that you have made without exiting from the panel.

- Click **OK** to apply the changes that you have made and exit from the panel.

Custom properties reference

You can set values for a number of custom properties. This reference section describes each of the custom properties.

- "General properties"
- "Custom properties for single sign-on protocol service"
- "Custom properties for the trust service" on page 448
- "Custom properties for the key service" on page 450
- "Custom properties for a SOAP client" on page 451
- "Custom properties for SAML 2.0" on page 451
- "Custom properties for the console" on page 451
- "Custom property for OpenID" on page 452
- "Custom property for transport security protocol" on page 452
- "Custom properties for LTPA tokens" on page 452

To add the custom properties to your domain configuration, see "Creating a custom property" on page 445.

General properties

DistributedMap.GetRetryLimit

When specified, and when the value is greater than 0, the wrapper will query the distributed map the configured number of times before returning that the data is not in the map.

- Value type: Integer
- Example value: 2

DistributedMap.GetRetryDelay

When the retry limit is higher than 1, this value sets the time to wait in milliseconds between retries. The default is 2000, or 2 seconds.

- Value type: Integer
- Example value: 2000

*componentName***.statisticsEnabled**

When specified as True, statistics tracking function for a specific component is enabled and the data collected and can be retrieved using the mechanisms presented by the component. When set to false, statistics are not tracked. Typically, this property is set to true for components that need to be timed or counted.

- Value type: Boolean
- Example value: False

Custom properties for single sign-on protocol service

requireSoapActionForSoap

This parameter controls the single sign-on protocol service behavior when it receives a request through the browser POST method and it needs to determine if it is a SOAPRequest or a BrowserRequest. Use of this parameter enables the service to handle non-compliant SOAP clients that do not send the required SOAPAction header on SOAP requests.

Default value: true

- Value type: boolean
- Example value: true

requireContentTypeForSoap

This parameter controls whether or not a SOAPRequest must contain a content-type of either `text/xml` or `application/soap+xml`. This parameter enables the single sign-on protocol service to handle non-compliant SOAP clients.

Note: When this parameter, and requestSoapActionForSoap are both false, all posts will be interpreted as SOAPRequests.

Default value: true

- Value type: boolean
- Example value: True

POC.allowsCredRefresh

When set to true, this parameter causes the LocalLogoutAction to be skipped on the service provider during single sign-on and federation. Instead, the credentials are refreshed. Set this parameter to true for the Web Plug-ins. Otherwise, set it to false.

Default value: true

- Value type: boolean
- Example value: True

SPS.PageFactory.HtmlEscapedTokens

A comma-separated list of tokens that must be HTML-escaped when being rendered in pages sent to the browser. Typically, this property includes any macros in the SPS.PageFactory.Exception2Macro runtime custom property (if used). This property is an important security consideration for preventing cross-site scripting vulnerabilities.

- Value type: string
- Example value: @TOKEN_A@, @TARGET@

SPS.PageFactory.Exception2Macro

This runtime custom property is a comma-separated list of classname:macro pairs. Classname is the full name of an exception class. Macro is the replacement macro to which the class maps. The macro must start and end with "@" as shown in the example values.

- Value type: string
- Example values: com.demo.MyException: @MYEXCEPTION@, com.tivoli.am.fim.trustserver.sts.STSException: @STSEXCEPTION@

SPS.POC.Default.Header.Names.Enabled

When specified, this property enables the use of default header names for the point of contact header values. If false, the only headers that will be read or written will have to be part of the sps.xml configuration file.

- Value type: boolean
- Example value: false

POC.WebSeal.SignOutInfoDelegate.UserSessionIdHeaderName

This value overrides the default tagvalue_user_session_id.

- Value type: String
- Example value: tagvalue_user_session_id

SPS.WebSealPoc.ContextPoolSize

Specifies the number of PDContext objects available in the pool. This value reflects the number of clients that need to be authorized when using single sign-on. You might need to increase the value based on the logout load of the system. When a large number of logouts occur at the same time, the Tivoli Federated Identity Manager runtime might run out of PDContext objects and logouts might start to fail. Because each PDContext object uses system resources, such as memory and file descriptors, care should be taken to select a value. The value must be greater than 0.

Default value: 5

- Value type: integer
- Example value: 5

SPS.WebSealPoc.DisablePDSignout

When set to true, this parameter disables the signout functionality of the single sign-on protocol service WebSEAL Point of Contact client. When the signout operation is invoked, it logs that no signout occurs and returns successfully. When this parameter is enabled, the single sign-on protocol service does not require the Tivoli Access Manager Java runtime (PDJRTE) to be configured.

Default value: false

- Value type: boolean
- Example value: true

SPS.WebSealPoc.Force.PdAdmin.Task

When set to true, this value forces the WebSeal Point of Contact callback to always use **pdadmin server** tasks to logout the user.

- Value type: boolean
- Example value: false

SPS.WebSealPoc.ContextPoolInitAttempts

This value represents the amount times that the PDContext objects initialization will be tried. The default is 1 and the value needs to be greater then 0.

- Value type: integer
- Example value: 1

SPS.WebSealPoc.ContextPoolInitTimeout

This value represents the maximun amount of time to be used during PDContext objects initialization. After the time has expired, the initialization will stop. The default is 10000 and the value needs to be greater then 0. The amount is on miliseconds.

- Value type: integer
- Example value: 10000

Custom properties for the trust service

username.disable.password.validation

When set to true, this parameter causes the UsernameTokenSTSModule to skip password validation.

The default is false.

- Value type: boolean
- Example value: true

username.jaas.provider.hostname

This parameter allows for specifying an alternate name for the local host in the event that WebSphere was not configured with the value of localhost for the host name.

The default is localhost.

- Value type: String
- Example value: localhost

username.jaas.provider.port

This parameter allows for specifying the port configured for the local WebSphere NameServer service.

The default is 2809.

- Value type: Integer
- Example value: 2809

pdjrte.context.min.pool.size

Specifies the minimum size of the Authorization context pool. This parameter is used by the UsernameTokenSTSModule. This parameter should be set only if recommended by a performance evaluation.

- Value type: Integer
- Example value: 5

pdjrte.context.max.pool.size

Specifies the maximum size of the Authorization context pool. This parameter is used by the UsernameTokenSTSModule. This parameter should be set only if recommended by a performance evaluation.

- Value type: Integer
- Example value: 50

ivcred.allow.groupUpdate

When set to true, will attempt to modify the credential by adding groups.

Note: Do not under any circumstances use this parameter.

- Value type: boolean
- Example value: false

saml.use.rst.lifetime

Directs the SAML modules to use the lifetime of the RequestSecurityToken element to derive the lifetime of the issued SAML assertion. When false, does not use the RST lifetime.

Default value: false

- Value type: boolean
- Example value: false

passticket.disable.uppercase.principal

Directs the PassTicket Module not to transform the principal name to all uppercase before attempting to generate a Passticket using the native RACF handler. When false, always raises the principal to uppercase for the native RACF handler.

Default value: false

- Value type: boolean
- Example value: false

sts.use.issuer.saml20.sso

The default is false, which directs the SAML 2.0 module to use the Issuer value, instead of the NameID NameQualifier value to look up an alias when performing a single sign-on operation.

Default value: false

- Value type: boolean
- Example value: false

username.wss.namespace.override

If not specified, the default is the WSS 1.1 token profile namespace. The key for this property can be used as a prefix to set the scope of the property to a specific STS Chain (i.e. username.wss.namespace.override.uuid1234)

- Value type: string
- Example value: <*a_URI_namespace*>

Custom properties for the key service

kessjksservice.include.keyinfo.x509.certificate.data

Includes a base64 encoded certificate in the KeyInfo element of the signature. When this element is true, either by default or by explicit use of this property, then the other KESS runtime properties are ignored. When not specified, the default is true.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.subject.key.identifier

Includes the subject key identifier in the KeyInfo element of the signature when the given certificate supports it. Can be used in addition to issuer.details and subject.name. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.issuer.details

Adds X509 issuer details to the KeyInfo element of the signature. Can be used in addition to subject.key.identifier and subject.name. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.subject.name

Adds the X509 subject distinguished name (DN) to the KeyInfo element of the signature. Can be used in addition to subject.key.identifier and issuer.details. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.exclude.inclusive.namespace.prefixes

A comma separated list of prefix names. When set the prefixes in the list will not be added to the InclusiveNamespaces list that is in the Signature Element.

- Value type: String
- Example value: ds

Custom properties for a SOAP client

com.tivoli.am.fim.soap.client.jsse.provider

The Java Secure Socket Extension (JSSE) provider name that should be used instead of IBMJSSE for SOAP client socket connections.

- Value type: String
- Example value: IBMJSSE

com.tivoli.am.fim.soap.client.jce.provider

The Java Cryptography Extension (JCE) provider name that should be used instead of IBMJCE for SOAP client keystores.

- Value type: String
- Example value: IBMJCE

com.tivoli.am.fim.soap.client.trust.provider

The Java Trust Manager provider algorithm name that should be used instead of IbmX509 for SOAP client Trust Managers.

- Value type: String
- Example value: IbmX509

Custom properties for SAML 2.0

SAML20.LogoutRequest.NotOnOrAfter.Enabled

When specified as true, the NotOnOrAfter attribute will be included on LogoutRequest messages from the identity provider to the service provider.

Default value: True

- Value type: Boolean
- Example value: True

SAML20.LogoutRequest.NotOnOrAfter.Lifetime

Specifies the time in seconds used to set the NotOnOrAfter attribute on a logout request.

Default value: 120

- Value type: Integer
- Example value: 300

Custom properties for the console

STS.showSSOChains

This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for single sign-on transactions. Note that setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.

- Value type: boolean
- Example value: true

STS.showUSCChains

This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for User Self Care federations. Note that setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.

- Value type: boolean
- Example value: true

STS.showAQChains

This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for SAML 2 federations that enable the Attribute Query service. Note that setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.

- Value type: boolean
- Example value: true

Custom property for OpenID

OpenID.TrustedSitesManagerModuleID

This is a plugin module id for a module that implements the com.tivoli.am.fim.protocols.openid_trusted_sites_manager extension point. There are two examples which implement this extension:

- TrustedSitesManagerCookieImpl
- TrustedSitesManagerMemoryImpl

When the parameter is not specified, the default is TrustedSitesManagerCookieImpl.

- Type: String
- Example value: TrustedSitesManagerCookieImpl

Custom property for transport security protocol

Specifying the transport security protocol for HTTPS connections

The default secure protocol for HTTPS connections created by IBM Tivoli Federated Identity Manager is `SSL_TLS`. To change (override) the default protocol, specify the following runtime custom property in the `fim.appservers.properties` file:

```
com.tivoli.am.fim.soap.client.ssl.protocol= PROTOCOL
```

where the value of *PROTOCOL* can be any of the following values: `SSL_TLS`, `SSL`, `SSLv2`, `SSLv3`, `TLS` or `TLSv1`.

Custom properties for LTPA tokens

Specifying custom Tivoli Federated Identity Manager runtime properties that force compatible QName generation

WebSphere Application Server versions 6.0.2 and 6.1 do not distinguish between LTPA v1 and LTPA v2 tokens in Web Services. Only one BinarySecurityToken ValueType is supported for LTPA tokens, and the QName of the value type is:

```
http://www.ibm.com/websphere/appserver/tokentype/5.0.2#LTPA
```

When the Tivoli Federated Identity Manager STS issues an LTPA v2 token, the token is created with the following QName. This QName is correct, but it is not supported by WebSphere Application Server versions 6.0.2 and 6.1:

```
http://www.ibm.com/websphere/appserver/tokentype#LTPAv2
```

This APAR provides custom Tivoli Federated Identity Manager runtime properties that force compatible QName generation if needed. To enable compatibility mode, set either or both of the following custom runtime properties:

```
ltpa.enable.compat.mode.[chainid_uuid]=true ltpa.enable.compat.mode=true
```

where *chainid_uuid* is the value of the Chain UUID. For example:

ltpa.enable.compat.mode.[uuideb42e428-011b-1ebc-a0cb-9e6c4b35c1c7]=true

To determine the value of Chain UUID, in the administration console select **Trust Service Chains**-> **Select Action**, then select **Show Chain ID in column in table**. This action selection causes a new column to appear in the table that displays the unique Chain ID.

Chapter 40. Customizing an authentication login form for single sign-on

Customize an authentication login form by adding parameters to a WebSphere or WebSEAL point of contact server profile.

When a user requests access to a single sign-on federation, the identity provider initiates single sign-on by authenticating the user. To authenticate the user, the identity provider uses a point of contact server to display a forms-based login page.

When an identity provider participates in multiple federations or hosts multiple partners in one federation, the administrator can customize the default login form.

As administrator, you can customize

- The login page based on the contents of the requests sent by the service providers.
- The look and feel of the login form.
- The type of authentication required.
- The login pages for WebSEAL and WebSphere point of contact servers.

To customize the login page, use the Tivoli Federated Identity Manager administration console to configure a new point of contact server profile. In the new profile, add a parameter to the authentication callback, and specify one or more values for the parameter.

Tivoli Federated Identity Manager provides some parameters which are always available and consistent across all federation types and some which are specific to the type of federation.

The protocols which support protocol-specific parameters are:

- SAML 1.x
- SAML 2
- OpenID

The set of defined values are described in "Supported macros for customizing an authentication login form"

Task overview:

1. Review the supported values for your protocol type, and identify the ones you want to use. See "Supported macros for customizing an authentication login form"
2. Create a new point of contact server profile. See "Configuring a point of contact server to support customization of login pages" on page 458

Supported macros for customizing an authentication login form

This topic describes the set of macros for customizing an authentication login form.

Tivoli Federated Identity Manager supplies contextual authentication parameters in customizing login forms. In the setting WebSEAL as the point of contact server, these are query-string parameters to the login page. For WebSphere, they are in the WASReqURL cookie when the login page is loaded. The parameters are macros in the configuration of the authentication callback for the point of contact server profile.

Note: When you use the WebSphere point of contact, the value of the query string parameter needs to be URL decoded twice.

Supported macros are:
- Protocol independent macros
- SAML protocol macros
- OpenID protocol macros

Note: If the value of `authentication.macros` is longer than the permitted length of query string parameter, the WASReqURL cookie will not be present in the identity provider.

Protocol independent macros for customizing an authentication login form

The following macros are protocol independent and can be used regardless of the federation type used.

Table 130. Supported Protocol independent macros

Macro	Query-String Parameter name	Description
%FEDID%	FedId	Specifies a unique identifier (UUID) used internally by Tivoli Federated Identity Manager to identify the federation.
%FEDNAME%	FedName	Specifies the user-assigned name of the federation.

SAML protocol supported macros for customizing an authentication login form

The following macros are supported for SAML protocol. Macros are supported for both SAML 1.x and SAML 2.0, except as indicated.

Table 131. Supported SAML protocol macros

Macro	Query-String Parameter name	Description
%PARTNERID%	PartnerId	Represents the SSO partner that the user is trying to sign in to. SAML value: The value is the ProviderID of the partner.

Table 131. Supported SAML protocol macros (continued)

Macro	Query-String Parameter name	Description
%TARGET%	Target	Represents the target URL at the partner, if known. SAML value: The value is the value of the target parameter.
%ACSURL%	AssertionConsumerURL	Represents the assertion consumer service URL of the partner, if applicable. SAML value: The value is the Partner ACS URL.
%AUTHNCONTEXT%	AuthnContext	**Supported for SAML 2.0 only** Represents the AuthnContext in request (if applicable). SAML value: The value is a base-64 encoded string representing the XML from the RequestedAuthnContext in the SAML AuthnRequest (if present).
%SSOREQUEST%	SSORequest	**Supported for SAML 2.0 only** Represents the entire SSO request (if applicable). SAML value: The value is a base-64 encoded string representing the XML from the entire SAML AuthnRequest.
%FORCEAUTHN%	ForceAuthn	**Supported for SAML 2.0 only** The value `true` or `false`. SAML value: If the ForceAuthn flag is set in the SAML 2 SSO request causing the user to re-authenticate, the value is `true`. Otherwise the value is `false`.

OpenID supported macros for customizing an authentication login form

The following macros are supported for the OpenID protocol.

Table 132. Supported OpenID protocol macros

Macro	Query-String Parameter name	Header
%PARTNERID%	PartnerId	Represents the SSO partner that the user is trying to sign in to. OpenID value: The value of the openid.trustroot parameter.
%TARGET%	Target	Represents the target URL at the partner, if known. OpenID value: The value of the openid.return_to parameter.
%SSOREQUEST%	SSORequest	Represents the entire SSO request (if applicable). OpenID value: The checkid_setup request as a base64-encoded version of the url-encoded SSO request.
%UNSATISFIEDPAPEPOLICIES%	UnsatisfiedPapePolicies	Represents a list of strings which represent PAPE policies. These strings are returned as "not yet satisfied" by the identity provider mapping rule in an OpenID identity provider federation. OpenID value: PAPE policies returned in the ContextAttributes Attribute openid.pape.to_be_satisfied_auth_policies
%FORCEAUTHN%	ForceAuthn	Specifies if authentication on the identity provider is forced. The values are `true` or `false`. OpenID value: The value is `true` if one of these criteria is satisfied: • the PAPE max_auth_age was zero (meaning forced to authenticate again) • the IDP mapping rule on the OpenID identity provider is forcing authentication due to unsatisfied PAPE policies • the authentication time returned by the IDP mapping rule does not satisfy the (non-zero) max_auth_age requested by the RP Otherwise the value is `false`.

Configuring a point of contact server to support customization of login pages

This topic describes how to a configure custom point of contact server to support customization of a login page.

Before you begin

Ensure that you:

- Understand how customized login pages are supported. See Chapter 40, “Customizing an authentication login form for single sign-on,” on page 455
- Know which macros to specify for the authentication callback parameter. See “Supported macros for customizing an authentication login form” on page 455

Note: You do not need to create and publish a custom Point of Contact callback plug-in before specifying authentication macros. Support for authentication macros is provided by default. When you run the configuration wizard, you can ignore the message that states that you must publish a plug-in before using the wizard.

About this task

The following procedure describes how to add a custom point of contact server that is like a point of contact server already defined in your environment in order to modify the information displayed in a login page.

Procedure

1. Log in to the administration console.
2. Click **Tivoli Federated Identity Manager** → **Domain Management** → **Point of Contact**
3. Select the existing point of contact server that you want to base your new point of contact server on. You must select a profile for either WebSEAL or **WebSphere**.
4. Click **Create Like** to display the Welcome Panel of the Point of Contact Profile wizard.
5. Click **Next** to display the Profile Name panel. It shows information from the profile on which you are basing your new point of contact server.
6. Type a name for the profile and an optional a description.
7. Click **Next** to display Sign in panel is displayed.
8. Accept the default entries for the sign-in callbacks, the parameters for each callback, and the order in which they are used and click **Next**.
9. Accept the default entries for the Sign out panel and click **Next**.
10. Accept the default entries for the Local ID panel and click **Next**.
11. Click **Add Parameters** in the Callback Parameters section on the Authentication panel.
12. Enter `authentication.macros` at Name.
13. Enter the macros you want to use at Values. To specify multiple values and separate the macros, place a backslash (\) and a comma between values. For example: %FEDID%\,%FEDNAME%\,%PARTNERID%
14. Click **Next** to display the Summary panel. It lists all the callbacks and parameters you specified in the preceding steps.
15. Click **Finish** to complete the setup or click **Back** to return to the previous panels and revise your selections.
16. Click **Current Domain portlet**.
17. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

What to do next

"Activating a point of contact server" on page 484

Chapter 41. Customizing single sign-on event pages

Tivoli Federated Identity Manager generates files that are displayed in response to events that occur during single sign-on requests. The response displayed might be a form (such as when login information is required) or an error or information statement about a condition that occurred while the request was processed.

You have the option of customizing the event pages, as follows:

- Modifying their appearance or content.
- Specifying which geographic or language locale to use when the pages are displayed.

Before continuing with the customization, you should have a thorough understanding of how event pages are generated and displayed. Refer to "Generation of event pages."

Generation of event pages

Event pages are displayed in response to events that occur during single sign-on requests. They usually contain a form (such as a prompt for user name and password information) or text (such as an informational or error message).

Event pages are dynamic pages that are generated by Tivoli Federated Identity Manager using the following information:

Template files
XML or HTML files that are provided with Tivoli Federated Identity Manager and contain elements, such as fields, text, or graphics, and sometimes macros that are replaced with information that is specific to the request or to provide a response to the request.

Page identifiers
Event information that corresponds to one or more template files. Each page identifier corresponds to a specific event condition, such as a specific error or a condition in which a message or a form must be displayed. To create an event page, page identifiers are mapped to one or more template files. The mapping function allows multiple page identifiers to point to the same template file.

Message catalogs
Text that is used to replace macros in the template files.

When a request is received, the appropriate response page is generated as follows:

1. Processing of the request occurs and a response to an event is required.
2. Template files and page identifiers are read from the file system.
3. Macros in the template files are replaced with values that are appropriate for the response that is needed.
4. An appropriate event page is generated.
5. The generated event page is displayed.

For information about the relationship between page identifiers and template files, see "Page identifiers and template files" on page 462.

Page identifiers and template files

A page identifier specifies an event and each event corresponds to one or more *template files*. Some page identifiers are specific to the specification (such as SAML 1.x) and some are general. To modify the elements (text, graphics, and so on) of the page that is displayed for an event, you will need to either modify the template file or copy a template file, use the copy as the basis for a new file, and then map the event to that new file.

General page identifiers and their template files

Table 133. General page identifiers and their template files

Page identifier (Event)	Description	Template file
/proper/errors/noprotdet	Displayed when protocol is unknown	/proper/errors/noprotdet.html
/proper/errors/missing_component	Displayed when protocol is unknown	/proper/errors/missingcomponent.html
/proper/errors/protocol_error	Displayed when a protocol module throws an exception	/proper/errors/protocol_error.html
/proper/errors/need_authentication	Displayed when the initial URL information is not found on the user session.	/proper/errors/need_authentication.html
/proper/errors/access_denied	Displayed when access is denied.	/proper/errors/access_denied.html
/proper/errors/missing-initial-url.html	Displayed when the initial URL information is not found on the user session.	/proper/errors/allerror.html
/proper/errors/unauth-access-to-waspoc-delegate.html	Displayed when the WebSphere point of contact delegate protocol has been accessed without proper authentication.	/proper/errors/allerror.html
/proper/login/formlogin.html	Displayed when using form-based authentication. **Attention:** Do not change the action value and parameter names for the form POST. They must remain unchanged for the form to function properly.	/proper/login/formlogin.html
/proper/login/formloginerror.html	Displayed when an error occurs using the formlogin.html file. See "Customizing the login form" on page 75.	/proper/login/formloginerror.html
/proper/genericpoc/login_success.html	Displayed when the generic point of contact implementation performs a successful login without a target URL.	/proper/login/login_success.html
/proper/waspoc/login_success.html	Displayed when the WebSphere point of contact implementation performs a successful login without a target URL.	/proper/login/login_success.html
/proper/waspoc/login_failure.html	Displayed when an error occurs during login using the WebSphere point of contact implementation.	/proper/login/login_failure.html

SAML 1.x page identifiers and their template files

Table 134. SAML 1.x page identifiers and their template files

Page identifier (Event)	Description	Template file
/saml/invalid_request.html	Displayed when a request is not valid.	/saml/allerror.html
/saml/unknown_sp.html	Displayed when an unknown service provider is encountered.	/saml/allerror.html
/saml/unknown_ip.html	Displayed when an unknown identity provider is encountered.	/saml/allerror.html
/saml/invalid_ip_request.html	Displayed when an identity provider provides an invalid request.	/saml/allerror.html
/saml/unauth_user.html	Displayed when the running user has not authenticated.	/saml/allerror.html
/saml/cannot_exchange_for_sp.html	Displayed when there is an error encountered during the token exchange.	/saml/allerror.html
/saml/no_ip_post_page.html	Displayed when the identity provider does not have a POST page.	/saml/allerror.html
/saml/no_return_token.html	Displayed when there is no return token.	/saml/allerror.html
/saml/ip_post_to_sp.html	Displays the POST HTML form when the identity provider posts the SAML response to the service provider.	/saml/allerror.html
/saml/invalid_response.html	Displayed when an invalid response message is encountered.	/saml/allerror.html
/saml/ip_response_invalid.html	Displayed when identity provider response is invalid.	/saml/allerror.html
/saml/cannot_exchange_for_resource.html	Displayed when there is an error encountered during the token exchange	/saml/allerror.html
/saml/missing_context_attribute.html	Displayed when the required context attribute is not represent.	/saml/allerror.html
/saml/missing_config_parameter.html	Displayed when a required SPS configuration item is missing.	/saml/allerror.html
/saml/could_not_retrieve_assertion.html	Displayed when the service provider could not get the assertion from the Response or from the SOAP back channel.	/saml/allerror.html
/saml/could_not_perform_local_auth.html	Displayed when an error is encountered when the EAI header is returned.	/saml/allerror.html
/saml/could_not_create_signed_request.html	Displayed when a signed SAML assertion request cannot be generated.	/saml/allerror.html
/saml/sp_missing_target.html	Displayed at the service provider if the initial request to the WAYF endpoint does not contain a TARGET parameter.	/saml/allerror.html

Table 134. SAML 1.x page identifiers and their template files (continued)

Page identifier (Event)	Description	Template file
/saml/ error_parsing_soap_response.html	Displayed when there is an error encountered when the service provider attempts to retrieve the Assertion from the identity provider's SOAP endpoint.	/liberty/ error_parsing_soap_response.html
/saml/unknown_ip_wayf.html	Displayed when the "where you are from" cookie contains an identity provider ID that is not configured on the federation.	/saml/allerror.html

SAML 2.0 page identifiers and their template files

Table 135. SAML 2.0 page identifiers and their template files

Page identifier	Description	Template files
/saml20/error_building_msg.html	Displayed for errors building SAML 2 messages.	/saml20/error_building_msg.html
/saml20/ error_missing_config_param.html	Displayed when an invalid configuration parameter is detected at runtime.	/saml20/ error_missing_config_param.html
/saml20/error_sending_msg.html	Displayed for errors sending SAML 2 messages.	/saml20/error_sending_msg.html
/saml20/error_validating_msg.html	Displayed for errors validating SAML 2 messages.	/saml20/error_validating_msg.html
/saml20/error_validating_art.html	Displayed for errors validating SAML 2 artifacts.	/saml20/error_validating_art.html
/saml20/invalid_msg.html	Displayed for errors validating SAML 2 messages.	/saml20/invalid_msg.html
/saml20/invalid_art.html	Displayed for errors validating SAML 2 artifacts.	/saml20/invalid_art.html
/saml20/authn_failed.html	Displayed when a SAML 2 authentication fails.	/saml20/authn_failed.html
/saml20/logout_failed.html	Displayed for logout failures.	/saml20/logout_failed.html
/saml20/art_exchange_failed.html	Displayed when exchange of a SAML artifact for a response fails.	/saml20/art_exchange_failed.html
/saml20/nimgmt_update_failed.html	Displayed for name identifier management update failure.	/saml20/nimgmt_update_failed.html
/saml20/ nimgmt_terminate_failed.html	Displayed for name identifier management terminate failure.	/saml20/ nimgmt_terminate_failed.html
/saml20/ error_validating_msg_signature.html	Displayed for errors validating SAML 2 message signatures.	/saml20/ error_validating_msg_signature.html
/saml20/error_decrypting_msg.html	Displayed for errors decrypting SAML 2 messages.	/saml20/error_decrypting_msg.html
/saml20/error_parsing_msg.html	Displayed for errors parsing SAML 2 messages.	/saml20/error_parsing_msg.html
/saml20/error_parsing_art.html	Displayed for errors parsing SAML 2 artifacts.	/saml20/error_parsing_art.html
/saml20/invalid_init_msg.html	Displayed for errors validating SAML 2 messages.	/saml20/invalid_init_msg.html

Table 135. SAML 2.0 page identifiers and their template files (continued)

Page identifier	Description	Template files
/saml20/ error_validating_init_msg.html	Displayed for errors validating SAML 2 messages.	/saml20/ error_validating_init_msg.html
/saml20/logout_success.html	Displayed for successful logouts.	/saml20/logout_success.html
/saml20/logout_partial_success.html	Displayed for partial logout completion.	/saml20/logout_partial_success.html
/saml20/ nimgmt_terminate_success.html	Displayed for name identifier management terminate success.	/saml20/ nimgmt_terminate_success.html
/saml20/ nimgmt_update_success.html	Displayed for name identifier management update success.	/saml20/ nimgmt_update_success.html
/saml20/consent_to_federate.html	Displayed to prompt a user for consent to federate.	/saml20/consent_to_federate.html
/saml20/saml_post_artifact.html	Displayed for sending SAML 2.0 artifacts for POST profiles.	/saml20/saml_post_artifact.html
/saml20/saml_post_request.html	Displayed for sending SAML 2.0 requests for POST profiles.	/saml20/saml_post_request.html
/saml20/saml_post_response.html	Displayed for sending SAML 2.0 responses for POST profiles.	/saml20/saml_post_response.html
/saml/ could_not_create_signed_request.html	Displayed when a signed SAML assertion request cannot be generated.	/saml/allerror.html
/saml/sp_missing_target.html	Used at the service provider if the initial request to the WAYF endpoint does not contain a TARGET parameter.	/saml/allerror.html

Liberty page identifiers

Table 136. Liberty page identifiers

Page identifier	Description
/liberty/error_parsing_soap_response.html	Reports that SOAP response could not be parsed.
/liberty/fed-terminate-success.html	Displayed when termination successful
/liberty/lib-cant-modify-alias.html	Displayed when alias modification fails.
/liberty/lib-fed-consent.html	Sent to ask user for consent to federate.
/liberty/lib-fed-post-request.html	Form used for POSTing an authentication request
/liberty/lib-fed-post.html	Form used for POSTing a response.
/liberty/lib-internal-error-page.html	Sent for an error if nothing else can be sent.
/liberty/lib-ipi-consent.html	Asks user to consent to perform IP introduction to the service providers.
/liberty/lib-ipi-post.html	Reports IP introduction success.
/liberty/lib-login-failed-page.html	Not currently used.
/liberty/lib-logout-failed-page.html	Sent to user by the IP when logout failed for any reason.
/liberty/lib-logout-page.html	Sent to user to report all session terminations after a logout.

Table 136. Liberty page identifiers (continued)

Page identifier	Description
/liberty/lib-logout-success-page.html	Sent to user by the IP to report a successful logout.
/liberty/logoutFailure.gif	Image to indicate logout failure if the HTTP GET single logout technique is being used.
/liberty/logoutSuccess.gif	Image to indicate logout success if the HTTP GET single logout technique is being used.
/liberty/lib-message-timestamp-failure.html	Sent if the issue time is outside tolerance.
/liberty/lib-no-fed-exists.html	Sent when no federation exists.
/liberty/lib-no-liberty-assertion.html	Reports no assertion found in response.
/liberty/lib-no-local-login.html	Reports failure of local login.
/liberty/lib-no-service-available.html	Reports no assertion or alias service exists.
/liberty/lib-register-name-identifier-success.html	Reports successful registration of a name identifier.
/liberty/lib-request-id-not-matching-resp.html	Reports that a response does not correlate to any known request.
/liberty/lib-sig-validation-failure.html	Not currently used.
/liberty/lib-version-mismatch.html	Not currently used.
/pages/itfim/wayf/wayf-html.html	HTML WAYF response.

WS-Federation page identifiers

Table 137. WS-Federation page identifiers

Page identifier	Description
/wsfederation/cannot_exchange_for_resource.html	Reports that IP's WS-Trust request failed on the service provider
/wsfederation/cannot_exchange_for_sp.html	Reports that IP could not exchange a token for the service provider.
/wsfederation/cannot_local_auth.html	Used when the service provider cannot validate a token
/wsfederation/invalid_ip_response.html	Reports that the service provider could not understand an IP response.
/wsfederation/invalid_request.html	Not a WS-Federation request.
/wsfederation/invalid_sp_request.html	Displayed when a request is not valid.
/wsfederation/ip_post_to_sp.html	Used by WS-Federation for sending information from the IP to the service provider
/wsfederation/no_ip_post_page.html	Displayed when the identity provider does not have a post page
/wsfederation/no_return_token.html	Reports that IP could not find a token to return to the service provider
/wsfederation/signout_cleanup_failed.html	Not currently used.
/wsfederation/signout_cleanup_failed_no_auth.html	Used when WS-Federation signout failed because the user was not authenticated.
/wsfederation/signout_cleanup_to_sp.html	Used by WS-Federation signout to trigger signouts on service providers.

Table 137. WS-Federation page identifiers (continued)

Page identifier	Description
/wsfederation/signout_successful.html	Used when WS-Federation signout is successful
/wsfederation/sp_ip_returned_fault.html	Reports that fault returned by IP to the service provider
/wsfederation/unauth_user.html	Reports that user not authenticated on this IP
/wsfederation/unknown_ip_wayf.html	Reports that the service provider could not determine IP.
/wsfederation/unknown_sp.html	Reports that the service provider is unknown to this IP.

Low-level independent page identifiers

Table 138. Independent page identifiers

Page identifier	Description
/proper/errors/cannot_process	Used for unspecified internal errors
/proper/errors/missing_component	Displayed when protocol is unknown
/proper/errors/noprotdet	Displayed when protocol is unknown
/proper/errors/not_started	Used when the SPS is not running, which usually indicates a configuration error of some type
/proper/errors/protocol_error	Displayed when a protocol module throws an exception
/pages/itfim/wayf/error-no-ips.html	Reports that no identity providers exist, so WAYF processing cannot be done
/pages/itfim/wayf/error-missing-template.html	Used when no WAYF template page can be found
/pages/itfim/wayf/error-invalid-template.html	Used when the WAYF page is invalid
/pages/itfim/wayf/wayf-html.html	Displayed when a federation has more than one identity provider and the ITFIM_WAYF_IDP query-string parameter or the WAYF cookie is not present.

Location of template files

The template files are stored in the following directory by default:

AIX

`/usr/IBM/FIM/pages/`*`locale`*`/`

HP-UX, Linux, or Solaris

`/opt/IBM/FIM/pages/`*`locale`*`/`

Windows

`C:\Program Files\IBM\FIM\pages\`*`locale`*`\`

The locale subdirectory is specific to the geographic or language locales of the template files. The default locale directory is named `C` and all files are in English. If a language pack was installed, additional locales are available.

The template files are published from their default subdirectories into WebSphere Application Server directories. See “Publishing updates” on page 473.

Attention: If you need to modify the template files, modify them on the Tivoli Federated Identity Manager server. *Do not* modify them in the WebSphere Application Server directories.

Content of template files

HTML template files can contain macros that are replaced with context-specific information that is retrieved when the response page is built and returned. If your template file contains, for example, the macro `@EXCEPTION_MSG@`, an exception message is included in the response page.

The presence of a macro in a template file does not guarantee that the macro will have an actual value when the response page is built. A value for the macro must be defined when the page is built in order for the macro to return a value.

When customizing an HTML template file, use only those macros defined in the template file. If you add new macros to the template file, values for the added macros will not be returned when the final response page is generated.

Macros use the following format:

```
@MACRO@
```

Where *MACRO* represents the name of the macro; for example, `@EXCEPTION_MSG@`

The following macros are used in the template files.

Table 139. Macros used in the template files

Substitution macro	Brief Description
@ACTION@	The action is the URL where the form that contains the POST response will be sent. Used in an HTML POST response sent by an identity provider to a browser for a single sign-on protocol service request.
@CAUSE@	Information regarding the cause of the error.
@DETAIL@	Additional information about an error or exception that has occurred as part of request processing. Because additional text is not always available, even if the @DETAIL@ macros is used in an HTML template file, there is no guarantee that the macros will provide additional text about the exception.
@EXCEPTION_MSG@	Text message that describes an exception that has occurred in request processing.
@EXCEPTION_STACK@	Full exception stack of an exception that has occurred in request processing.
@FEDERATION_DISPLAY@	The name of the current federation, that is, the one currently in use.
@FEDERATION_ID@	The unique identifier of the current federation.
@PARTNER_ID@	Federation single sign-on protocol of a federation partner.
@REQ_ADDR@	Internet protocol (IP) address of the endpoint that requested a federation action.

Table 139. Macros used in the template files (continued)

Substitution macro	Brief Description
@RESPONSE@	Used in an HTML POST response from an identity provider, substituted for with the SAML response.
@SAMLSTATUS@	Collection of SAML status values received during the single sign-on action processing.
@SOAP_ENDPOINT@	The SOAP endpoint URL that is used to retrieve the assertion using a SAML artifact.
@TARGET@	Used to provide the service provider target in an HTML POST response sent by an identity provider) to a browser for a single sign-on protocol service request.
@TIMESTAMP@	A value for the current time.
@TOKEN:form_action@	The URL where the form that contains the POST message will be sent during a POST binding.
@TOKEN:IPDisplayName@	The identity provider unique name.
@TOKEN:IPProviderID@	The identity provider unique identifier.
@TOKEN:PartnerID@	The partner unique identifier.
@TOKEN:RelayState@	The SAML protocol RelayState value.
@TOKEN:SamlMessage@	The base64 encoded SAML message that is sent on a form.
@TOKEN:SPDisplayName@	The service provider unique name.
@TOKEN:SPProviderID@	The service provider unique identifier.
@TOKEN:UserName@	The authenticated user name that submitted the single sign-on action.
@WAYF_FEDERATION_DISPLAY_NAME@	Display name of the current federation, as presented in the console. Used on a page presenting a WAYF (Where Are You From) challenge and requesting that a user choose an identity provider.
@WAYF_FEDERATION_ID@	Identifier of the current federation in the configuration file. Used on a page presenting a WAYF challenge and requesting that a user choose an identity provider.
@WAYF_FORM@	Identifier information for the WAYF HTML form that is presented to a user to acquire identity provider information in a SPS action where the identity provider for the requestor has not yet been determined (it is not yet in the cookie presented).
@WAYF_FORM_ACTION@	Endpoint of the single sign-on protocol service; this should be the originally requested address (URL). Used on a page presenting a WAYF challenge and requesting that a user choose an identity provider.
@WAYF_FORM_METHOD@	HTTP method used on a request that has resulted in a WAYF on a page that is prompting for identity provider information. The method can be either GET, POST or HEAD.
@WAYF_FORM_PARAM_ID@	Identifier of the form parameter for the current identity provider, will typically be the configured cookie name. Used on a page that is presenting a WAYF challenge and requesting that a user choose an identity provider.

Table 139. Macros used in the template files (continued)

Substitution macro	Brief Description
@WAYF_HIDDEN_NAME@	Name of one of the initial parameters to a request that results in a WAYF, and is used on a page that is prompting for identity provider information.
@WAYF_HIDDEN_VALUE@	Value of one of the initial parameters to a request that results in a WAYF, and is used on a page that is prompting for identity provider information.
@WAYF_IP_DISPLAY_NAME@	The configured display name of the current identity provider on a page presenting a WAYF challenge.
@WAYF_IP_ID@	Configuration ID of the current identity provider on a page presenting a WAYF challenge.

Template page for the WAYF page

The Where Are You From (WAYF) page is used at the service provider. The WAYF page enables users to select their identity provider if there is more than one configured in the federation.

When a user arrives at a service provider, a WAYF identifier can be delivered through a cookie or query-string parameter with the request. The entity ID of the identity provider is stored as the value of the cookie or query-string parameter. If the WAYF identifier cookie or query-string parameter is not present, the WAYF page is displayed.

An example URL that includes the query string parameter for WAYF:

```
https://sp.host.com/FIM/sps/samlfed/saml20/
logininitial?RequestBinding=HTTPRedirect&ResponseBinding
=HTTPPost&ITFIM_WAYF_IDP=https://idp.host.com/FIM/sps/samlfed/saml20
```

This example is for a SAML 2.0 single sign-on URL. The query string parameter name is `ITFIM_WAYF_IDP`. The value of the identity provider ID is `https://idp.host.com/FIM/sps/samlfed/saml20`.

The WAYF page requires the user to indicate where they came from. If the user is not logged in to their identity provider, they are asked to log in. Depending on the attributes passed, the service provider can grant or deny access to the service.

The template pages are stored in the following directory by default:

<FIM_Install_Dir>`/pages/`*<locale>*`/pages/itfim/wayf`

See, “Low-level independent page identifiers” on page 467 for more information about WAYF template pages.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

This template file provides several replacement macros:

@WAYF_FORM_ACTION@
This macro is replaced with the endpoint of the original request. This macro does not belong within a repeatable section.

@WAYF_FORM_METHOD@
This macro is replaced with the HTTP method of the original request. This macro does not belong within a repeatable section.

@WAYF_FORM_PARAM_ID@
This macro is replaced with ID used by the action for the identity provider. This macro is repeated once for each identity provider.

@WAYF_IP_ID@
This macro is replaced with the unique ID of the identity provider. This macro is repeated once for each identity provider.

@WAYF_IP_DISPLAY_NAME@
This macro is replaced with the configured display name of the identity provider. This macro is repeated once for each identity provider.

@WAYF_HIDDEN_NAME@
This macro is replaced with the name of the hidden parameter. This macro is repeated once for each original request parameter and is hidden.

@WAYF_HIDDEN_VALUE@
This macro is replaced with the value of the hidden parameter. This macro is repeated once for each original request parameter and is hidden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!--
html wayf template that presents the choice as radio buttons.
-->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>Where are you from</title>
    </head>
    <body style="background-color:#ffffff">
        <div>
            <!--
            Insert the federation ids here just so we can show some tokens
            [RPT federations]
                @WAYF_FEDERATION_ID@
                @WAYF_FEDERATION_DISPLAY_NAME@
            [ERPT federations]
            -->
            <form id="wayfForm" name="wayfForm"
                    action="@WAYF_FORM_ACTION@" method="@WAYF_FORM_METHOD@">
                <div>
                    <table>

                    [RPT ips]
                        <tr>
                            <td>
                                <input type="radio" id="@WAYF_FORM_PARAM_ID@"
                                    name="@WAYF_FORM_PARAM_ID@"
                                    value="@WAYF_IP_ID@"/>@WAYF_IP_DISPLAY_NAME@
                            </td>
                        </tr>
                    [ERPT ips]

                    </table>
                    <!-- the hidden inputs must be present -->

                    [RPT hidden]
                    <input type="hidden" name="@WAYF_HIDDEN_NAME@"
                        id="@WAYF_HIDDEN_NAME@"
                        value="@WAYF_HIDDEN_VALUE@"/ >
                    [ERPT hidden]

                </div>
                <input type="submit" name="submit" value="Submit" />
            </form>
        </div>
    </body>
</html>
```

Figure 55. Template page wayf-html.html

Modifying or creating the template files

To customize the appearance of the event pages, you can modify the template files or create new template files.

Before you begin

Before continuing with this procedure, be sure that you are familiar with how event pages are generated. See "Generation of event pages" on page 461.

About this task

Attention: Modify the template files in the directory on the Tivoli Federated Identity Manager server (as described below). When all of your changes are complete, publish them to the WebSphere Application Server configuration repository directory. *Do not* edit these files in the configuration repository.

To modify or create new template files:

Procedure

1. Decide which event pages you want to modify. Refer to the list of events and their corresponding template files at "Page identifiers and template files" on page 462.
2. Stop the WebSphere Application Server where the runtime component is installed. Use the stopServer command. Refer to the WebSphere Information Center if you need help.
3. Locate the template file that corresponds to the event page you want to modify or make a copy of an existing template file and use it create a new file. The template files are located in a geographic-specific or language-specific locale subdirectory for the file. The default locale subdirectory is named `C` and all files are in English. If a language pack was installed, additional locales are available. You can also create your own locales, as described in "Creating a page locale" on page 474. The default directory for the template files is as follows:

   AIX
   `/usr/IBM/FIM/pages/`*`locale`*`/`

   HP-UX, Linux or Solaris
   `/opt/IBM/FIM/pages/`*`locale`*`/`

   Windows
   `C:\Program Files\IBM\FIM\pages\`*`locale`*`\`
4. Use a text editor to modify or create new files.
5. Save the files to the appropriate location, such as the same directory where you edited them or from which you copied them.

What to do next

When you have completed this step, continue with publishing the files to the configuration repository as described in "Publishing updates."

Publishing updates

When all updates and additions have been made to the template files, you must publish the files to the configuration repository so they will be displayed.

About this task

To publish updates to the event pages:

Procedure

1. Log in to the management console.
2. In the console, click **Tivoli Federated Identity Manager** → **Domain Management** → **Event Pages**.
3. Locate the event or events that you want to map to the new or updated pages.
4. In the **HTML Page Displayed** field for each event you are modifying, type the path and file name for the file you want to use for that event.
5. Click **Apply**. A warning message is displayed that explains you must publish the updated files to the configuration repository.
6. Click **Publish Pages** to publish the changes right away. Otherwise, click **Close** and later, when you are ready to publish click **Domain Management** → **Runtime Node Management** and on the Runtime Node Management panel, click the **Publish pages** button.

Creating a page locale

The template files that are used to generate event pages are located in a geographic-specific or language specific locale subdirectory for the file. The default locale subdirectory is named `C` and all files are in English. Additional locales and corresponding languages are also available. In addition, you can create your own locale.

Before you begin

Before continuing with this procedure, be sure that you are familiar with how event pages are generated. See "Generation of event pages" on page 461.

About this task

To create your own locale:

Procedure

1. Log in to the management console.
2. In the console, click **Tivoli Federated Identity Manager** → **Domain Management** → **Event Pages**. The Event Pages panel is displayed.
3. Click the **Page Locale** tab to open the Page Locale panel.
4. Click **Create**. A placeholder list item is added to the list of page locales with the Page Locale name of **locale** and a Page Root Directory of **page_root**.
5. Enter a locale abbreviation to replace **locale**.
6. Enter a name for the directory of the locale in place of **page_root**.
7. Click **Apply** or **OK**. A warning message is displayed that explains you must publish the updated files to the configuration repository.
8. Click **Publish** to publish the changes right away. Otherwise, click **Close** and later, when you are ready to publish click **Domain Management** → **Runtime Node Management** and on the Runtime Node Management panel, click the **Publish pages** button.

Deleting a page locale

You can delete any page locales other than the default C page locale, which was installed when Tivoli Federated Identity Manager was installed.

About this task

Deleting a page locale removes it from the environment and prevents the pages in that locale from being displayed.

Procedure

1. Log in to the management console.
2. In the console, click **Tivoli Federated Identity Manager** → **Domain Management** → **Event Pages.**. The event page is displayed
3. Click the **Page Locale** tab to open the Page Locale panel.
4. In the **Select** field, select the button next to the page locale you want to remove. For descriptions of the locales, refer to the online help.
5. Click **Delete** and then click **Apply** to apply your changes and remain in the Page Locale portlet or click **OK** to apply your changes and exit from the portlet.

Customizing multiple-use physical page templates

In certain circumstances you might need to customize physical page templates that are referred to by multiple page identifiers.

About this task

There are some physical page templates that are referred to by multiple page identifiers in `sps.xml`.

For example, `<sps:PageIdentifierMapping name="/liberty/error_parsing_soap_response.html" location="/liberty/error_parsing_soap_response.html" />` and `<sps:PageIdentifierMapping name="/saml/error_parsing_soap_response.html" location="/liberty/error_parsing_soap_response.html" />`

If the SAML response needs to differ from the Liberty response, you will need to edit the pages as follows:

Procedure

1. For each locale affected, copy the Liberty page into the SAML directory.
2. Edit the two pages as desired.
3. Edit the second PageIdentifierMapping above to read `<sps:PageIdentifierMapping name="/saml/error_parsing_soap_response.html" location="/saml/error_parsing_soap_response.html" />`
4. Publish these changes as described in “Publishing updates” on page 473

Customizing the Consent to Federate Page for SAML 2.0

A *consent to federate page* is an HTML form which prompts a user to give consent in joining a federation. You can customize the *consent to federate page* to specify what information it requests from a user.

Before you begin

Determine what values you want to use for the consent to federate page. See About this task for a list of the values.

About this task

When a user accesses a federation, they agree to join it. The HTML form `consent_to_federate.html` prompts for this consent. You can customize what the form requests by adding consent values. These values indicate how a user agrees to join a federation and if service providers are notified of the consent. Identity providers receive the consent values in the SAML 2.0 response.

The following values determine how a user joins a federation:

1 A user agrees to join a federation without notifying the service provider.

0 A user refuses to join a federation

A URI value
A URI can indicate whether the user agrees to join a federation and if you want to notify the service provider about the user consent. The following table lists and describes the supported URI values.

Table 140. Supported consent values for SAML 2.0 response

Consent value	URI	Description
Unspecified	`urn:oasis:names:tc: SAML:2.0:consent: unspecified`	The consent of the user is not specified.
Obtained	`urn:oasis:names:tc: SAML:2.0:consent: obtained`	Specifies that user consent is acquired by the issuer of the message.
Prior	`urn:oasis:names:tc: SAML:2.0:consent: prior`	Specifies that user consent is acquired by the issuer of the message before the action which initiated the message.
Implicit	`urn:oasis:names:tc: SAML:2.0:consent: current-implicit`	Specifies that user consent is implicitly acquired by the issuer of the message when the message was initiated.
Explicit	`urn:oasis:names:tc: SAML:2.0:consent: current-explicit`	Specifies that the user consent is explicitly acquired by the issuer of the message at the instance that the message was sent.
Unavailable	`urn:oasis:names:tc: SAML:2.0:consent: unavailable`	Specifies that the issuer of the message was not able to get consent from the user.
Inapplicable	`urn:oasis:names:tc: SAML:2.0:consent: inapplicable`	Specifies that the issuer of the message does not need to get or report the user consent.

Follow these steps to customize the consent to federate page:

Important: Modify the template files on the Tivoli Federated Identity Manager server. When all of your changes are complete, you can publish them on the WebSphere® Application Server configuration repository directory. **Do not** edit these files in the configuration repository.

Procedure

1. Use the `stopServer` command to stop the WebSphere Application Server where Tivoli Federated Identity Manager is installed. For more information, see WebSphere Information Center.

2. Use a text editor to access `consent_to_federate.html`.

   The template files are in a geographic-specific or language-specific subdirectory. All files are in English. If you installed a language pack, additional locales are available. The default directory depends on the operating system.

   AIX® `/usr/IBM/FIM/pages/locale/saml20/`

   HP-UX, Linux®, or Solaris
   `/opt/IBM/FIM/pages/locale/saml20/`

   Windows®
   `C:\Program Files\IBM\FIM\pages\locale\saml20\`
3. Add the appropriate consent values for your federation. See About this task for a complete list of values.
4. Save the files to the appropriate location. This location might be same directory where you edited them.
5. Restart the WebSphere Application Server.

Example

The following example shows an added URI with a consent value `Obtained`:

```
<input type="radio" checked name="Consent"
value="urn:urn:oasis:names:tc:SAML:2.0:consent:obtained"/>
Consent Obtained.br/>
```

In this example, the user consent is acquired by the issuer of the message.

What to do next

Publish the files to the configuration repository. See “Publishing updates” on page 473.

Chapter 42. Developing a custom point of contact server

The point of contact server in your Tivoli Federated Identity Manager environment is the first entity to process a request for access to a resource. You can choose one of the provided options for a point of contact server or you can create a custom point of contact server.

About this task

A custom point of contact server is made up of several customized callback modules that define sign in, sign out, local ID, and authentication. A custom point of contact server might be appropriate in your environment if you want to integrate an existing authentication or Web access management application with Tivoli Federated Identity Manager. For example, a custom point of contact server would be useful in the following scenarios:

- If you have an existing single sign-on cookie token that is used throughout your existing enterprise, you could implement a custom point of contact server that uses a SignIn callback that sets that custom single sign-on domain cookie that conforms to your existing single sign-on strategy.
- If you have an existing Web access management product that exposes a custom API for asserting a user identity to the environment or retrieving the current user for the request. You could implement a point of contact server that uses a local identity callback (to retrieve the user for the transaction) or implement a custom point of contact server that uses a SignIn callback to assert the user identity to the environment, or implement a point of contact server that uses both types of callbacks.

Developing a custom point of contact server requires programming experience with developing callback modules and knowledge of Tivoli Federated Identity Manager programming concepts. Refer to the developerWorks links in the information center at http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp.

When you have completed the development work, you will need to integrate the solution with your Tivoli Federated Identity Manager environment as follows:

Procedure

1. Publish the callback plug-ins to the Tivoli Federated Identity Manager runtime module. See “Publishing callback plug-ins” on page 480.
2. Gather the parameter information that you will need for configuring each of the callback modules.
3. Create a new point of contact server profile. You have the option of creating a new profile or using an existing profile as the basis for your new point of contact server profile. See either:
   - “Creating a new point of contact server” on page 480
   - “Creating a point of contact server like an existing server” on page 482
4. Activate the point of contact server. See “Activating a point of contact server” on page 484.

Publishing callback plug-ins

If you have developed the modules for a custom point of contact server, you must publish the plug-ins for those modules so that you can use them in your Tivoli Federated Identity Manager environment.

Before you begin

Before continuing with this task, ensure that you have developed the appropriate callback modules for your custom point of contact server. For more information, refer to Chapter 42, “Developing a custom point of contact server,” on page 479.

Procedure

1. Copy the callback plug-ins to the /plugins directory where you installed Tivoli Federated Identity Manager. For example, on Windows, the directory is `/opt/IBM/FIM/plugins`.
2. Log in to the console and click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Runtime Node Management**.
3. The Runtime Node Management panel is displayed. Click **Publish Plug-ins**.

What to do next

After publishing the plug-ins, you can continue with creating the point of contact profile.

Creating a new point of contact server

Tivoli Federated Identity Manager provides several options for a point of contact server depending on your role in the federation. In addition, you have the option of developing your own point of contact server. If you have developed your own, you must add it to your environment using the console.

Before you begin

Before you can add the custom point of contact server to your environment, you must:

- Publish any custom point of contact callback plug-ins to the runtime node. Refer to “Publishing callback plug-ins.”
- Know what type of parameters will be used, if any, and the corresponding values that need to be passed to these callbacks.

About this task

The following procedure describes how to add a custom point of contact server that is unlike any point of contact server already defined in your environment. If you are adding a custom point of contact server that is similar to another point of contact server, use the procedure in “Creating a point of contact server like an existing server” on page 482.

Procedure

1. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.
2. Click **Create**. The Welcome panel of the Point of Contact Profile wizard is displayed.

3. Ensure that you have completed the prerequisite steps. Then, click **Next**. The Profile Name panel is displayed.
4. Type a name for the profile of your custom point of contact server and optionally a description. Click **Next**. The Sign In panel is displayed.
5. In the Sign In panel, you will specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
   a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
   b. Click the **Add Parameters** button. A callback parameters section is displayed for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are displayed.
   c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
   d. Repeat the preceding steps until all parameters have been added to all the callbacks.
6. Click **Next**. The Sign Out panel is displayed.
7. In the Sign Out panel, you will specify the sign-out callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
   a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
   b. Click the **Add Parameters** button. A callback parameters section is displayed for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are displayed.
   c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
   d. Repeat the preceding steps until all parameters have been added to all the callbacks.
8. Click **Next**. The Local ID panel is displayed.
9. In the Local ID panel, you will specify the local ID callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
   a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
   b. Click the **Add Parameters** button. A callback parameters section is displayed for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are displayed.
   c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
   d. Repeat the preceding steps until all parameters have been added to all the callbacks.
10. Click **Next**. The Authentication panel is displayed.

11. In the Authentication panel, you will specify the sign-out callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
    a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
    b. Click the **Add Parameters** button. A callback parameters section is displayed for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are displayed.
    c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
    d. Repeat the preceding steps until all parameters have been added to all the callbacks.
12. Click **Next**. The Summary panel is displayed. It lists all the callbacks and parameters you specified in the preceding steps.
13. Click **Finish** to complete the setup or click **Back** to return to previous panels and revise your selections.

What to do next

To make this point of contact server active, continue with the instructions in "Activating a point of contact server" on page 484.

Creating a point of contact server like an existing server

Tivoli Federated Identity Manager provides several options for a point of contact server depending on your role in the federation. In addition, you have the option of developing your own point of contact server and basing it on an existing server. If you have developed your own, you must add it to your environment using the console.

Before you begin

Before you can add the custom point of contact server to your environment, you must:

- Publish any custom point of contact callback plug-ins to the runtime node. Refer to "Publishing callback plug-ins" on page 480.
- Know what type of parameters will be used, if any, and the corresponding values that need to be passed to these callbacks.

About this task

The following procedure describes how to add a custom point of contact server that is like a point of contact server already defined in your environment. If you are adding a custom point of contact server that is not similar to an existing point of contact server, use the procedure in "Creating a new point of contact server" on page 480.

Procedure

1. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.

2. Select the existing point of contact server that you will base your new point of contact server on.
3. Click **Create Like**. The Welcome panel of the Point of Contact Profile wizard is displayed.
4. Ensure that you have completed the prerequisite steps. Then, click **Next**. The Profile Name panel is displayed and the information from the profile that you are basing your new point of contact server on is displayed.
5. Type a name for the profile of your custom point of contact server and optionally a description. Click **Next**. The Sign In panel is displayed.
6. In the Sign In panel, you will specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback. Because you selected a profile for this point of contact server to be based on, the callbacks and parameters for that profile will be displayed as the ones in use. If you need to add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
7. Click **Next**. The Sign Out panel is displayed.
8. In the Sign Out panel, you will specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback. Because you selected a profile for this point of contact server to be based on, the callbacks and parameters for that profile will be displayed as the ones in use. If you need to add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
9. Click **Next**. The Local ID panel is displayed.
10. In the Local ID panel, you will specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback. Because you selected a profile for this point of contact server to be based on, the callbacks and parameters for that profile will be displayed as the ones in use. If you need to add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
11. Click **Next**. The Authentication panel is displayed.
12. In the Authentication panel, you will specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback. Because you selected a profile for this point of contact server to be based on, the callbacks and parameters for that profile will be displayed as the ones in use. If you need to add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
13. Click **Next**. The Summary panel is displayed. It lists all the callbacks and parameters you specified in the preceding steps.
14. Click **Finish** to complete the setup or click **Back** to return to previous panels and revise your selections.

What to do next

To make this point of contact server active, continue with the instructions in “Activating a point of contact server” on page 484.

Activating a point of contact server

To enable a point of contact server as the active server in your environment, you must activate it.

Procedure

1. Log in to the console. Click **Tivoli Federated Identity Manager** → **Manage Configuration** → **Point of Contact**.
2. Select the point of contact server you want to activate.
3. Click **Make Active**. The point of contact server you selected is activated and will be used as the point of contact server in your environment.

Chapter 43. Customizing signature X.509 certificate settings

When you sign messages or assertions, the X.509 certificate (public key) is included with your signature as a base64-encoded X.509 certificate. However, you have the option of specifying whether this data should be excluded and whether additional data should be included with your signatures.

Before you begin

Before using this procedure, you must have configured your federation. In addition, if you are an identity provider in a SAML 1.x federation, your assertion signature settings are configured when you add your service provider partners. To modify the settings for your assertion signature, you must have already configured a service provider partner.

About this task

To modify your signature settings:

Procedure

1. Log in to the console and click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**. Or, if you are an identity provider to modify your SAML 1.x assertion signature settings, click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Partners**
2. The Federations panel displays a list of configured federations. Select a federation. The Partners panel displays a list of configured partners Select a partner.
3. Click **Properties**.
4. Select the properties to modify. The properties are described in the online help.
5. When you have finished modifying properties, click **OK** to close the Properties panel.

Chapter 44. Running WebSphere Application Server with Java 2

If you are running Java 2 security on the WebSphere Application Server where Tivoli Federated Identity Tivoli Federated Identity Manager is installed, you must modify the java.policy to grant permission to the Tivoli Federated Identity directories that are in the temp subdirectory of your WebSphere profile.

About this task

To modify the java.policy:

Procedure

1. Locate the java.policy directory and open it in a text editor. The default locations of the file are:

   On AIX:
   /usr/IBM/WebSphere/AppServer/java/jre/lib/security/java.policy

   On HP-UX, Linux, or Solaris:
   /opt/IBM/WebSphere/AppServer/java/jre/lib/security/java.policy

   On Windows:
   C:\Program Files\IBM\WebSphere\AppServer
2. Add the following lines to java.policy:

   ```
   grant codeBase "file:${server.root}/temp/node_name/server_name/
     ITFIMManagementService/-" {permission java.security.AllPermission;
   };
   grant codeBase "file:${server.root}/temp/node_name/server_name/
     ITFIMRuntime/-" {permission java.security.AllPermission;
   };
   ```

   node_name is the name of the node such as IBM-FCFB36CC28ENode05

   server_name is the name of the server such as server1
3. Save and close the java.policy file.
4. Restart WebSphere Application Server.

Part 7. Appendixes

Appendix A. tfimcfg reference

The tfimcfg command can be used to configure LDAP settings for the Integrated Solutions Console installation, and also to configure WebSEAL as a Point of Contact server.

tfimcfg usage

```
TFIM Autoconfiguration Tool Version 6.1.0 [060316a]

Usage: java -jar tfimcfg.jar [-action <mode>] [options]
The tfimcfg tool has several modes of operation.  Each mode uses different
command line options.

Configuring and unconfiguring WebSEAL servers:
   -action tamconfig: configures a WebSEAL server.  This mode is the default.
   Options:
      -cfgfile <file>: WebSEAL configuration file.
         This option is required.
      -rspfile <file>: response file for non-interactive configuration.
         Default: interactive configuration

   -action tamunconfig: unconfigures a WebSEAL server.
   Options:
      -cfgfile <file>: WebSEAL configuration file.
         This option is required.
      -rspfile <file>: response file for non-interactive unconfiguration.
         Default: interactive configuration

Configuring and unconfiguring LDAP servers:
   -action ldapconfig: configures an LDAP server.
      -rspfile <file>: response file to control the configuration.  The
         response file should be based on the sample ldapconfig.properties
         file.  This option is required.

   -action ldapunconfig: unconfigures an LDAP server.
      -rspfile <file>: response file to control the configuration.  The
         response file should be based on the sample ldapconfig.properties
         file.  This option is required.
```

When the tfimcfg tool is run to configure an LDAP server, the tool also creates several user accounts. The user accounts are needed by the single sign-on demonstration application.

When you run **tfimcfg** to set up the LDAP accounts for the administration console user, you call tfimcfg with the parameters:

```
-action ldapconfig
```

This action creates the demonstration user accounts.

tfimcfg limitation with Sun Java 1.4.2.4

Certain versions of Sun Java are incompatible with tfimcfg.

The incompatibility causes the following error:

```
HPDAZ0602E Corrupted file: Insufficient information to contact Policy Server
```

The problem occurs because the Sun JRE is unable to read the keystores generated by the Tivoli Access Manager PDJrteCfg. When this error occurs, you should either use an IBM JVM or else apply the latest JRE patches from Sun. If the problem persists after applying the patches from Sun, use an IBM JVM for the configuration.

tfimcfg LDAP properties reference

The tfimcfg utility reads a properties file to obtain the values to use when configuring an LDAP user registry. The properties file contains values that you can modify.

ldap.hostname
The LDAP server host name. Default: `localhost`

ldap.port
The LDAP port number. Default: `389`

The default value is for non-SSL communication. When you have configured the LDAP server to communicate using SSL, the default port is 636.

ldap.suffix.add
Boolean value that specifies whether tfimcfg adds suffixes to the LDAP server as needed. Supports IBM Tivoli Directory Server Versions 6.1, 6.0 and 5.2 only.

Default:

```
ldap.suffix.add=true
```

ldap.suffix.user.configuration
ldap.organization.configuration
Boolean values that specify whether tfimcfg creates LDAP containers to store Tivoli Federated Identity Manager users and groups. The Tivoli Federated Identity Manager users and groups are:

- Tivoli Federated Identity Manager server users and groups
- Tivoli Federated Identity Manager Installation Verification Tool (IVT) users and groups

When you do not need those users and groups, or you already have LDAP containers that you will use for those users and groups, set these values to `false`.

When ldap.organization.configuration is true, tfimcfg creates the `dc=example,dc=com` LDAP objects.

Default:

```
ldap.suffix.user.configuration=true
ldap.organization.configuration=true
```

ldap.suffix.alias.configuration
Boolean value that specifies whether tfimcfg creates an LDAP suffix to store single sign-on aliases. The default alias is `cn=itfim`.

```
ldap.suffix.alias.configuration=true
```

ldap.suffix.tam.configuration
Boolean value that specifies whether tfimcfg creates the `secAuthority=Default` suffix for Tivoli Access Manager.

- When you have already configured Tivoli Access Manager set this value to `false`.
- When Tivoli Access Manager is not using this LDAP server, set this value to `false`.

```
ldap.suffix.tam.configuration=true
```

Note: If the secAuthority=Default suffix already exists, the tfimcfg program ignores the value of the ldap.suffix.tam.configuration property.

ldap.fim.configuration
Boolean value that specifies whether tfimcfg configures LDAP for the Tivoli Federated Identity Manager alias service.

Default value: `true`.

ldap.ivt.sp.configuration
Boolean value that specifies whether tfimcfg creates users and groups for the service provider in the Installation Verification Tool (IVT) application.

Default value: `true`.

ldap.ivt.ip.configuration
Boolean value that specifies whether tfimcfg creates users and groups for the identity provider in the Installation Verification Tool (IVT) application.

Default value: `true`.

ldap.modify.acls
Boolean value that specifies whether tfimcfg attaches appropriate ACLs (access control lists) to the LDAP server. These ACLs grant read and write access to the Tivoli Federated Identity Manager administrative users created by tfimcfg.

Note that tfimcfg attaches ACLs for IBM LDAP and Sun ONE servers. For other LDAP servers, you must attach the ACLs manually.

When this is set to `false`, you must attach the ACLs manually.

Default value: `true`.

ldap.admin.dn
The DN used by the LDAP administrator to issue bind requests.

Default: `cn=root`

ldap.admin.password
The password for the LDAP administrator.

Default: `passw0rd`

ldap.security.enabled
Boolean value that specifies whether communication with the LDAP server must use SSL.

Default: `false`.

ldap.security.trusted.jks.filename
The name of the Java keystore that contains the signer of the LDAP-presented SSL certificate that LDAP presents during trusted communications.

ldap.suffix.user.dn
ldap.suffix.user.name
ldap.suffix.user.attributes
ldap.suffix.user.objectclasses
When you want tfimcfg.jar to create LDAP containers for your users, you can set these values to control the Distinguished Names (DNs) that are used.

Defaults:

```
ldap.suffix.user.dn=dc=com
ldap.suffix.user.name=com
ldap.suffix.user.attributes=dc
ldap.suffix.user.objectclasses=domain
```

ldap.suffix.alias.dn

Distinguished Name (DN) to use for storing single sign-on alias. This value of this property must begin with `cn=`. Modify this value when you do not want to use the default DN.

Default:

```
ldap.suffix.alias.dn=cn=itfim
```

ldap.organization.dn
ldap.organization.name
ldap.organization.attributes
ldap.organization.objectclasses

When you want tfimcfg.jar to create LDAP containers for your groups, you can set these values to control the Distinguished Names (DNs) that are used.

Defaults:

```
ldap.organization.dn=dc=example,dc=com
ldap.organization.name=example
ldap.organization.attributes=dc
ldap.organization.objectclasses=domain
```

ldap.user.container.dn
ldap.group.container.dn

The distinguished names to use for the containers for users and groups.

Defaults:

```
ldap.user.container.dn=cn=users,dc=example,dc=com
ldap.group.container.dn=cn=groups,dc=example,dc=com
```

ldap.fim.server.bind.dn
ldap.fim.server.bind.shortname
ldap.fim.server.bind.password

The distinguished name, short name, and password that the Tivoli Federated Identity Manager server (application) uses to bind to the LDAP server.

Default:

```
ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
ldap.fim.server.bind.shortname=fimserver
ldap.fim.server.bind.password=passw0rd
```

ldap.fim.admin.group.dn
ldap.fim.admin.group.shortname

The distinguished name and short name for the Integrated Solutions Console administration group.

Default:

```
ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
ldap.fim.admin.group.shortname=fimadmins
```

ldap.user.objectclasses
ldap.group.objectclasses
ldap.user.shortname.attributes

The values for LDAP containers for user objectclass, group objectclass, and user shortname attributes.

Default:

```
ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
ldap.group.objectclasses=groupOfUniqueNames
ldap.user.shortname.attributes=cn,sn,uid
```

Default ldapconfig.properties file

The ldapconfig.properties file is distributed as part of the runtime and management service component. Many properties have default values.

```
ldap.hostname=localhost
ldap.port=389

# If true, new suffixes will be added to the LDAP server as needed.
# Only supported for IDS 5.2 and 6.0
ldap.suffix.add=true

# If true, data for the LDAP user suffix (dc=com, by default) will be
# created.
ldap.suffix.user.configuration=true

# If true, data for the SSO alias suffix (cn=itfim, by default) will be
# created.
ldap.suffix.alias.configuration=true

# If true, create the secAuthority=Default suffix for TAM
ldap.suffix.tam.configuration=true
ldap.fim.configuration=true
ldap.ivt.sp.configuration=true
ldap.ivt.ip.configuration=true
ldap.organization.configuration=true
ldap.modify.acls=true

ldap.admin.dn=cn=root
ldap.admin.password=passw0rd

ldap.security.enabled=false
ldap.security.trusted.jks.filename=

ldap.suffix.user.dn=dc=com
ldap.suffix.user.name=com
ldap.suffix.user.attributes=dc
ldap.suffix.user.objectclasses=domain

# DN to use for storing SSO aliases.  This must begin with cn=
ldap.suffix.alias.dn=cn=itfim

ldap.organization.dn=dc=example,dc=com
ldap.organization.name=example
ldap.organization.attributes=dc
ldap.organization.objectclasses=domain

ldap.user.container.dn=cn=users,dc=example,dc=com
ldap.group.container.dn=cn=groups,dc=example,dc=com

ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
ldap.fim.server.bind.shortname=fimserver
ldap.fim.server.bind.password=passw0rd

ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
ldap.fim.admin.group.shortname=fimadmins

ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
ldap.group.objectclasses=groupOfUniqueNames
ldap.user.shortname.attributes=cn,sn,uid
```

Figure 56. Default values for ldapconfig.properties

Sample output from tfimcfg configuration of LDAP

The following figure shows sample output from the running of tfimcfg.

The command for running tfimcfg to configure LDAP entries for the alias service and the demonstration application is:

```
java -jar tfimcfg.jar -action ldapconfig -rspfile
   /tmp/ldapconfig.properties
```

Here is sample output from running the command on an identity provider. The example uses an ldapconfig.properties file that has the default values.

```
Configuring LDAP server.
LDAP server vendor: International Business Machines (IBM),
   version 6.0.
Adding LDAP suffix secAuthority=Default.
Reloading IBM Directory Server configuration.
Adding LDAP suffix dc=com.
Reloading IBM Directory Server configuration.
Creating LDAP object dc=com.
Adding LDAP suffix cn=itfim-cmd.
Reloading IBM Directory Server configuration.
Creating LDAP object cn=itfim-cmd.
Creating LDAP object dc=example,dc=com.
Creating LDAP object cn=users,dc=example,dc=com.
Creating LDAP object cn=groups,dc=example,dc=com.
Creating LDAP object uid=fimserver,cn=users,dc=example,dc=com.
Creating LDAP object cn=fimadmins,cn=groups,dc=example,dc=com.
Adding user uid=fimserver,cn=users,dc=example,dc=com to group
   cn=fimadmins,cn=groups,dc=example,dc=com.
Creating LDAP object o=identityprovider,dc=com.
Creating LDAP object cn=MEemployee,o=identityprovider,dc=com.
Creating LDAP object cn=MEmanager,o=identityprovider,dc=com.
Creating LDAP object cn=MEexecutive,o=identityprovider,dc=com.
Creating LDAP object cn=elain,o=identityprovider,dc=com.
Creating LDAP object cn=mary,o=identityprovider,dc=com.
Creating LDAP object cn=chris,o=identityprovider,dc=com.
Updating IBM LDAP ACLs for suffix CN=ITFIM-CMD.
Updating IBM LDAP ACLs for suffix SECAUTHORITY=DEFAULT.
Updating IBM LDAP ACLs for suffix DC=COM.
Done updating LDAP server configuration.
```

Figure 57. Sample output from tfimcfg.jar

Modifying the Object Class of Users Created by tfimcfg Utility

The **tfimcfg** utility, when invoked with the argument **-action ldapConfig** creates a set of demonstration users in LDAP. The object classes of these *demo* users are, however, incompatible with WebSphere's default search parameters for user entries in IBM Tivoli Directory Server. The demonstration mapping rules assume that this set of demonstration users is available in LDAP.

The **tfimcfg** utility creates user entries in LDAP with these object classes: `person,organizationalPerson,inetOrgPerson` . WebSphere's search parameters for IBM Tivoli Directory Server, however, require that user entries contain objectclass `ePerson`. Due to this mis-match of object class, the demonstration users cannot be located by WebSphere in the user registry.

A workaround for this situation is to modify the object classes of users created by the tfimcfg utility.

To add this object class to the list of object classes:

1. In a text editor, open the ldapconfig.properties file:

```
/opt/IBM/FIM/tools/tamcfg/ldapconfig.properties
```

2. Locate the following line:

   ```
   ldap.user.objectclasses=person,organizationalPerson,
     inetOrgPerson
   ```

3. Modify the line to read:

   ```
   ldap.user.objectclasses=person,ePerson,organizationalPerson,
     inetOrgPerson
   ```

4. Invoke tfimcfg -action ldapConfig.

To view a sample result of this change, use the following command:

```
# idsldapsearch -D cn=root -w passw0rd -b dc=com uid=mary
cn=mary,o=identityprovider,dc=com
displayName=Mary Manor
mail=mmanor@identityprovider.example.com
uid=Mary
userPassword=abcd1234
objectclass=top
objectclass=person
objectclass=ePerson
objectclass=organizationalPerson
objectclass=inetOrgPerson
employeenumber=987-65-4321
sn=Manor
cn=Mary
```

Appendix B. URLs for initiating SAML single sign-on actions

The SAML specifications provide limited or no guidance about the endpoints or methods that end users must use to initiate single sign-on actions. However, in a Tivoli Federated Identity Manager environment, URLs are defined that end user can use to initiate single sign-on actions.

The following reference is useful for architects or application developers who are implementing the user interaction components of their federation.

Note: These URLs are not used for partner-to-partner communication. For more information, see Chapter 14, “SAML endpoints and URLs,” on page 145.

SAML 1.x initial URL

The intersite transfer service URL is where the sign-on request process begins in a SAML 1.x federation. The URL for initiating a single sign-on request has the following syntax:

Syntax

```
https://identity_provider_hostname:port_number/sps/
  federation_name/samlxx/login?TARGET=
  service_provider_id/target_application_location
  [optional query strings]
```

Elements

https *or* **http**
The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

identity_provider_hostname
The hostname of the identity provider's point of contact server.

port_number
The port number of the intersite transfer service endpoint. The default value is 9443.

sps
The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml*xx*
The designation of the SAML protocol you choose to use in your federation. The values can be one of the following:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login
This element indicates what type of endpoint is using the port. **login** is used for the intersite transfer service.

You must also use the **TARGET** query string and you have the option of using either, both, or neither of the optional query strings (**SP_PROVIDER**) and (**PROTOCOL**), as follows:

TARGET
The URL of the target application that a user can log in to using single sign-on.

SP_PROVIDER_ID
The value of query string specifies the provider ID of the service provider that is the target of the single sign-on request. This query string is optional but recommended. The use of this query string removes any ambiguity about which service provider is the target of the single sign-on request. Without this query string, the service provider is determined by matching the *`URI://hostname[:port]`* of the URL in the TARGET query string to the *`URI://hostname[:port]`* of the provider ID for the service provider partner that is configured for the federation. This parameter is used with requests that are initiated at the identity provider.

PROTOCOL
The value of this parameter specifies the type of single sign-on profile (browser artifact or browser POST) that should be used for the single sign-on request. The syntax of the extension is `PROTOCOL=[BA|POST]`, with BA indicating Browser Artifact and POST indicating Browser POST. The query string overrides local identity provider configuration. The use of the extension is optional. When the extension is not present, the profile choice is determined by the configuration file settings. To enable use of this extension, you must enable the IBM PROTOCOL extension setting during the configuration steps for creating a SAML 1.x federation on an identity provider.

These query strings can be used individually or in combination. For example, the URL used to initiate single sign-on, when the SP_PROVIDER_ID is used but the PROTOCOL extension is not, has the following syntax:

```
https://intersite_transfer_service_URL?SP_PROVIDER_ID=
  provider_ID_of_service_provider&TARGET=target_application_URL
```

With the SP_PROVIDER_ID and the PROTOCOL extension, the URL has the following syntax:

```
https://intersite_transfer_service_URL?SP_PROVIDER_ID=
  provider_ID_of_service_provider&TARGET=target_application_URL
  &PROTOCOL=[BA|POST]
```

Examples

Single sign-on URL, without the optional parameters:
The following example shows the single sign-on URL for an identity provider using a federation named `ipfed`, the SAML 1.1 protocol, a service provider with a provider ID of `https://sp.example.com:9443`, and an application called `snoop`:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?TARGET=
  https://sp.example.com:9443/snoop/
```

Single sign-on URL, when SP_PROVIDER_ID and PROTOCOL extension *are* used:

The following example shows a URL that is used to initiate single sign-on when the IBM PROTOCOL extension *is* used. In this example, even if the identity provider is configured to use a POST profile for the service provider named `sp`, the following use of the PROTOCOL extension would force the identity provider to use the browser artifact profile:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?SP_PROVIDER_ID=
  https://sp.example.com:9443/sps/spfed/saml11&TARGET=
  https://sp.example.com:9443/
  snoop&PROTOCOL=BA
```

Single sign-on URL, when SP_PROVIDER_ID is used but the PROTOCOL extension is *not* used:

The following example shows a URL that is used to initiate single sign-on when the SP_PROVIDER_ID is used but the IBM PROTOCOL extension is *not* used:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?SP_PROVIDER_ID=
  https://sp.example.com:9443/sps/spfed/saml11&TARGET=
  https://sp.example.com:9443/snoop
```

SAML 2.0 profile initial URLs

The SAML 2.0 specification defines the endpoints that are to be used for partner-to-partner communications but it does not define the way in which end users can initiate a single sign-on action using those endpoints.

In a Tivoli Federated Identity Manager environment, specially formed URLs that incorporate the single sign-on action to take, the binding to be used for the action, and the location where the action should take place can be used for user-initiated single sign-on actions. These URLs are referred to as *profile initial URLs*.

Architects and application developers, who will design and implement their users' interaction with the single sign-on process, will need to understand profile initial URLs and incorporate them into their Web applications.

The following sections describe the format of the SAML 2.0 profile initial URLs that are supported in a Tivoli Federated Identity Manager environment.

Assertion consumer service initial URL (service provider)

In a SAML 2.0 federation, the assertion consumer service URL can be initiated at the identity provider server site or the service provider site. This topic describes the syntax for initiating single sign-on at the service provider.

Syntax for initiating single sign-on at the service provider

```
https://provider_hostname:port_number/sps/
federation_name/saml20/logininitial?
RequestBinding=RequestBindingType&
ResponseBinding=ResponseBindingType&
NameIdFormat=NameIDFormatType&
IsPassive=[true|false]&
ForceAuthn=[true|false]&
AllowCreate=[true|false]&
AuthnContextClassRef = ClassReference&
AuthnContextDeclRef = DeclarationReference&
AuthnContextComparison = [exact| minimum | maximum |better]&
Target=target_application_location
```

Elements

https *or* **http**

The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname

The hostname of the provider point of contact server.

port_number
The port number of the intersite transfer service endpoint. The default value is 9443.

sps The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml20
The designation of SAML 2.0.

logininitial
This element indicates what type of endpoint is using the port. **logininital** is used to initiate the single sign-on service.

The following query strings must also be used in the URL:

RequestBinding
The binding that is used to send the request. The valid values when initiating single sign-on at the service provider are:

- HTTPPost
- HTTPArtifact
- HTTPRedirect

ResponseBinding
The binding that is used by the responder to return the response. The valid values when initiating single sign-on at the service provider are:

- HTTPPost
- HTTPArtifact

Target The URL of the application that a user can log in to using single sign-on.

NameIdFormat
The name ID format that is to be used for name identifiers. Valid values are:

- Transient (anonymous)
- Persistent
- Encrypted (for encrypted name IDs)
- E-mail

Persistent is the default setting. If the NameIdFormat attribute is not included, a persistent name ID is used.

AllowCreate
Indicates if new persistent account linkage is performed on the request. The default value is `true`. **Note:** To use this parameter, the **NameIdFormat** must be set to Persistent.

ForceAuthn
Specifies whether the identity provider authenticates the user or not. A value of `true` means that the user must be authenticated. The default value is `false`.

Note:

- Depending on the federation configuration, the more restrictive setting is implemented. For example, if you set the federation configuration to force a user to authenticate, setting the ForceAuthn element to `false` is not implemented.
- If you plan to use WebSEAL cookie management with SAML 2.0 ForceAuthn, ensure that the list of managed cookies does not include the WebSphere session cookie. See "Configuring WebSEAL to manage cookies" on page 391

IsPassive

Indicates if the identity provider must take control of the user agent if set to `true`. The identity provider is not permitted to request the user to provide login credentials.

The default value is `false`.

Note: Depending on the federation configuration, the more restrictive setting is implemented. For example, if you set the federation configuration not to allow the identity provider to take control of the user agent, setting the IsPassive element to `false` is not implemented.

AuthnContextClassRef

Specifies one or more string values which identify authentication context class URI references.

Note: Use either `AuthnContextClassRef` or `AuthnContextDeclRef`. If both are supplied, `AuthnContextClassRef` is used.

AuthnContextDeclRef

Specifies one or more string values which identify authentication context declaration URI references.

Note: Use either `AuthnContextClassRef` or `AuthnContextDeclRef`. If both are supplied, `AuthnContextClassRef` is used.

AuthnContextComparison

Specifies the type of comparison used to determine the requested context classes or declarations. The comparison type must be one of the following:

- exact
- minimum
- maximum
- better

The default value is exact.

Example

Single sign-on URL when initiated at service provider:

The following example shows the single sign-on URL when initiated at a service provider. The name of the federation is `spfed`, and uses the SAML 2.0 protocol, HTTPPost as the request binding and response binding, and a target application at https://sp.example.com:9443/banking:

```
https://sp.example.com:9443/sps/
spfed/saml20/logininitial?
RequestBinding=HTTPPost&
ResponseBinding=HTTPPost&
NameIdFormat=persistent&
IsPassive=true&
ForceAuthn=true&
```

```
AllowCreate=true&
RequestedAuthnContext Comparison=minimum&
AuthnContextClassRef=classref1&
Target=https://sp.example.com:9443/banking
```

Single sign-on service initial URL (identity provider)

In a SAML 2.0 federation, the single sign-on service URL can be initiated at the identity provider server site or the service provider site. This topic describes the syntax for initiating the service at the identity provider.

Syntax for initiating single sign-on at the identity provider

```
https://provider_hostname:port_number/sps/
  federation_name/saml20/logininitial?RequestBinding=RequestBindingType&
  &PartnerId=target_partner_provider_ID
  &NameIdFormat=NameIDFormatType&AllowCreate=[true|false]
```

Elements

https *or* **http**
The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname
The hostname of the provider's point of contact server.

port_number
The port number of the intersite transfer service endpoint. The default value is 9443.

sps
The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml20
The designation of SAML 2.0.

logininitial
This element indicates what type of endpoint is using the port. **logininital** is used to initiate the single sign-on service.

The URL must also contain the following query strings:

RequestBinding
The binding that is used to send the response to the service provider. The valid values when initiating single sign-on at the identity provider are:

- HTTPPost
- HTTPArtifact

PartnerId
The provider ID of the target partner.

NameIdFormat
The name ID format that is to be used for name identifiers. Valid values are:

- Transient (anonymous)
- Persistent
- Encrypted (for encrypted name IDs)
- Email

Persistent is the default setting. If the NameIdFormat attribute is not included, a persistent name ID is used.

AllowCreate
Indicates if new persistent account linkage should be performed on the request. False is the default value. **Note:** To use this parameter, the **NameIdFormat** must be set to Persistent.

Example

Single sign-on URL when initiated at identity provider:
The following example shows the single sign-on URL when initiated at an identity provider in a federation named `ipfed`, using the SAML 2.0 protocol, HTTPPOST as the request binding and a partner ID of https://sp.example:9443/sps/saml20ip2/saml20. No NameIdFormat is specified, so a persistent name ID is used:

```
https://idp.example.com:9443/sps/ipfed/saml20/logininitial?
  RequestBinding=HTTPPost&
  PartnerId=https://sp.example:9443/sps/saml20ip2/saml20
```

Single logout service initial URL

In a SAML 2.0 federation , the single logout service URL is used by a partner to contact the Single logout profile. The URL to initiate the service has the following syntax:

Syntax

```
https://provider_hostname:port_number/sps/
  federation_name/saml20/sloinitial
..?RequestBinding=RequestBindingType
```

Elements

https *or* **http**
The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname
The hostname of the service or identity provider's point of contact server.

port_number
The port number of the artifact resolution service endpoint. The default value is 9444.

sps The designation for the Tivoli Federated Identity Manager server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml20
The designation that SAML 2.0 is used in your federation.

sloinitial
This element indicates what type of endpoint is using the port. **sloinitial**is used to initiate the single logout service

The following query must also be included:

RequestBinding
The binding that is used to send the request. The valid values are:

- HTTPPost

- HTTPRedirect
- HTTPArtifact
- HTTPSOAP

Examples

Single logout URL when initiated at service provider:

The following example shows the single logout URL when initiated at a service provider in a federation named `spfed`, using the SAML 2.0 protocol, HTTPRedirect as the request binding:

```
https://sp.example.com:9443/sps/spfed/saml20/sloinitial?
  RequestBinding=HTTPRedirect
```

Single logout URL when initiated at identity provider:

The following example shows the single logout URL when initiated at an identity provider in a federation named `ipfed`, using the SAML 2.0 protocol, HTTPArtifact as the request binding:

```
https://idp.example.com:9444/sps/ipfed/saml20/sloinitial?
  RequestBinding=HTTPArtifact
```

Name identifier management service initial URL

In a SAML 2.0 federation, the name identifier management service URL is used by a partner to contact the Name Identifier Management service. The URL to initiate the service has the following syntax:

Syntax

```
https://provider_hostname:port_number/sps/
  federation_name/mnidsinitial?RequestBinding=RequestBindingType
  &PartnerId=target_partner_provider_ID&NameIdTerminate=[True|False]
```

Elements

https *or* **http**

The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname

The hostname of the service or identity provider's point of contact server.

port_number

The port number of the artifact resolution service endpoint. The default value is 9444.

sps

The designation for the Tivoli Federated Identity Manager server. This element cannot be changed.

federation_name

The name you assign to the federation when you create it.

saml20

The designation that SAML 2.0 is used in the federation.

mnidsinitial

This element indicates what type of endpoint is using the port. **mnidsinitial** is used to initiate the name identifier.

The following query strings must also be included:

RequestBinding

The binding that is used to send the request to the partner. The valid values when initiating single sign-on at the identity provider are:

- HTTPPost
- HTTPArtifact
- HTTPRedirect
- HTTPSOAP

PartnerId

The provider ID of the target partner.

NameIdTerminate

A value that indicates if the name ID management flow should terminate the name ID mapping. Valid values are:

True Ends the account linkage.

False Indicates that the name ID flow will update the name identifiers (aliases). False is the default, if no value is explicitly specified.

Examples

Name identifier initiated at the identity provider:

The following example shows the name identifier URL initiated at an identity provider in a federation named `ipfed`, using the SAML 2.0 protocol and HTTP SOAP as the request binding:

```
https://idp.example.com:9444/sps/ipfed/saml20/mnidsinitial?
  RequestBinding=HTTPSOAP&PartnerId=https://saml20sp:444/sps/
  saml20/saml20&NameIdTerminate=true
```

Name identifier initiated at the service provider:

The following example shows the name identifier URL initiated at a service provider in a federation named `spfed`, using the SAML 2.0 protocol and HTTP Artifact as the request binding:

```
https://sp.example.com:9444/sps/spfed/saml20/mnidsinitial?
  RequestBinding=HTTPArtifact&PartnerId=https://saml20ip/FIM/sps/
  saml20/saml20&NameIdTerminate=true
```

Appendix C. Disabling logging to enhance performance

When using Tivoli Federated Identity Manager with Tivoli Access Manager, you can improve performance on a service provider by disabling logging for theTivoli Access Manager policy server.

To reduce usage of the central processor unit (CPU), complete the following steps:

1. Back up the policy director directory. For example, on Linux or UNIX:
   ```
   /opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector
   ```
2. Open the following file in a text editor:
   ```
   /opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector/PDJLog.properties
   ```
3. Disable message logging by setting the following parameter to false:
   ```
   baseGroup.PDJMessageLogger.isLogging=false
   ```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd
3-3-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Adobe, Acrobat, PostScript® and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel®, Intel logo, Intel Inside®, Intel Inside logo, Intel® Centrino®, Intel Centrino logo, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium®, and Pentium® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Glossary

alias service
The Tivoli Federated Identity Manager component that manages aliases, or name identifiers, that are passed between secure domains.

artifact
In the context of the SAML protocol, a structured data object that points to a SAML protocol message.

artifact resolution service
In the context of the SAML protocol, the endpoint in a federation where artifacts are exchanged for assertions.

assertion
In the context of the SAML protocol, data that contains authentication or attribute information or both types of information in a message.

assertion consumer service
In the context of the SAML protocol, the endpoint in a federation that receives assertions or artifacts as part of a single sign-on request or response.

binding
In the context of SAML, the communication method used to transport the messages.

browser artifact
A profile (that is, a set of rules) in the SAML standard that specifies that an artifact is exchanged to establish and use a trusted session between two partners in a federation. Contrast with *browser POST*.

browser POST
A profile (that is, a set of rules) in the SAML standard that specifies that a self-posting form be used to establish and use a trusted session between two partners in a federation. Contrast with *browser artifact*.

domain
A deployment of the Tivoli Federated Identity Manager runtime component on WebSphere Application Server.

endpoint
The ultimate recipient of an operation.

federation
A relationship in which entities, such as differing businesses, agree to use the same technical standard (such as SAML or Liberty), which enables each partner in the relationship to access resources and data of the other. See also identity provider and service provider.

identity mapping
The process of modifying an identity that is valid in an input context to an identity that is valid in an output context.

identity provider
A partner in a federation that has responsibility for authenticating the identity of a user.

intersite transfer service
In the context of the SAML protocol, the endpoint in a federation to which a single sign-on request is sent.

metadata
Data that describes a particular piece of information, such as settings for a configuration.

point of contact server
In the context of a federation, a proxy or application server that is the first entity to process a request for access to a resource.

profile
In the context of the SAML specification, a combination of protocols, assertions, and bindings that are used together to create a federation and enable federated single sign-on.

protocol
In the context of the SAML specification, a type of request message and response message that is used for obtaining authentication data and for managing identities.

SAML See *security assertion markup language*.

security assertion markup language
A set of specifications written by the OASIS consortium to describe the secure handling of XML-based request and

response messages that contain authorization or authentication information.

service provider
A partner in a federation that provides services to the user.

Simple and Protected GSS API Negotiation Mechanism (SPNEGO)
An authentication mechanism that provides single sign-on capability in Microsoft Windows environments.

single sign-on
An authentication process in which a user can access more than one system or application by entering a single user ID and password.

SOAP A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet.

SOAP back channel
Communications that take place directly between two SOAP endpoints.

SPNEGO
Simple and Protected GSS API Negotiation Mechanism

token A particular message or bit pattern that signifies permission or temporary control to transmit over a network. In the context of SAML, token is used interchangeably with *assertion*.

trust service
The Tivoli Federated Identity Manager component that manages security tokens that are passed between security domains. The trust service is also referred to as the *Security Token Service*.

Web service
A self-contained, self-describing modular application that can be published, discovered, and invoked over a network using standard network protocols. Typically, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available.

Web service security management
The Tivoli Federated Identity Manager component that is used to establish and manage federation relationships for Web service applications running on WebSphere Application Server that use WS-Security tokens.

Index

Printed in USA

GC27-2719-00