IBM® Tivoli® Federated Identity Manager
Version 6.2.2

Administration Guide

SC23-6191-02

IBM® Tivoli® Federated Identity Manager
Version 6.2.2

Administration Guide

SC23-6191-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 315.

Edition notice

Note: This edition applies to version 6, release 2, modification 2 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 6.2.2 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 administer 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.

Access to publications and terminology

This section provides:

- A list of publications in the IBM Tivoli Federated Identity Manager library.
- Links to "Online publications" on page xii.
- A link to the "IBM Terminology website" on page xii.

IBM Tivoli Federated Identity Manager library

The following documents are available in the IBM Tivoli Federated Identity Manager library:

- *IBM Tivoli Federated Identity Manager Quick Start Guide*
- *IBM Tivoli Federated Identity Manager Installation Guide*, GC27-2718-01
- *IBM Tivoli Federated Identity Manager Configuration Guide*, GC27-2719-02
- *IBM Tivoli Federated Identity Manager Installing, configuring, and administering risk-based access*, SC27-4445-02
- *IBM Tivoli Federated Identity Manager Configuring web services security*, GC32-0169-04
- *IBM Tivoli Federated Identity Manager Administration Guide*, SC23-6191-02
- *IBM Tivoli Federated Identity Manager Auditing Guide*, GC32-2287-05
- *IBM Tivoli Federated Identity Manager Troubleshooting Guide*, GC27-2715-01
- *IBM Tivoli Federated Identity Manager Error Message Reference*, GC32-2289-04

Online publications

IBM posts product publications when the product is released and when the publications are updated at the following locations:

IBM Tivoli Federated Identity Manager Information Center
The http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tspm.doc_7.1/welcome.html site displays the information center welcome page for this product.

IBM Security Systems Documentation Central and Welcome page
IBM Security Systems Documentation Central provides an alphabetical list of all IBM Security Systems product documentation and links to the product information center for specific versions of each product.

Welcome to IBM Security Systems Information Centers provides and introduction to, links to, and general information about IBM Security Systems information centers.

IBM Publications Center
The http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss site offers customized search functions to help you find all the IBM publications you need.

IBM Terminology website

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

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 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/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html

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:

- Tivoli Access Manager Information Center:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.itame.doc/toc.xml
- IBM WebSphere Application Server Version 8.0 Information Center:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/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:

- *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/td/tdprodlist.html

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 hard copies of some publications.

Many countries provide an online ordering service.

Follow these steps to access this service:

1. Go to http://www-947.ibm.com/support/entry/portal/Documentation
2. Select **IBM Publications Center** from **Getting Started**.
3. Select your country from **Select a country/region/language to begin** and click the arrow icon.
4. Follow the instructions for how to order hard copy publications on Welcome to the IBM Publications Center.

If your country does not provide an online ordering service, contact your software account representative to order publications.

Follow these steps to find your local contact:

1. Go to http://www.ibm.com/planetwide/
2. Click your country name to display a list of contacts.

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/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html.

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*.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

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.

Chapter 1. Using the console

IBM Tivoli Federated Identity Manager provides a management console that you can use to accomplish most administration tasks and some configuration tasks.

The IBM Tivoli Federated Identity Manager console is implemented as a plug-in to the IBM Integrated Solutions Console. The Integrated Solutions Console is a graphical administration console that provides a framework for administering multiple products. For example, you can administer IBM Tivoli Federated Identity Manager and WebSphere Application Server in the console.

Topics:

Logging in and logging out

Use the console to log in and log out of the Integrated Solutions Console. Access the console by opening a web browser and typing the appropriate URL.

About this task

To form the appropriate URL, you must know the settings that you used when you installed and configured the console. For example, the URL you need might be: `http://idp.example.com:9060/ibm/console`

This URL is made up of:

- The name of the host system that runs the console. For example:
  `http://idp.example.com`
- The port number of the Integrated Solutions Console.
  The default port for a console where WebSphere Application Server administration security is not enabled (http) is `9060`.
  The default port for a console where WebSphere Application Server administration security is enabled (https) is `9043`.
- The URL for accessing the console login page. This part of the URL is always the same:
  `/ibm/console`

When you have established the correct URL, you need the administrator account name and password for the console. The name and password depend on the version of WebSphere Application Server that was installed with IBM Tivoli Federated Identity Manager.

Version of WebSphere Application Server	Default administrator name
Existing version	The administrator name and password that were configured on WebSphere Application Server when IBM Tivoli Federated Identity Manager was installed.
Embedded version	`fimadmin` is the default administrator name. The password is assigned during installation.

Procedure

1. Enter the console URL in the address bar of your browser window. For example to the URL for a system with a host name of `idp.example.com`, using the default port number on a secured port, enter:

   `http://idp.example.com:9060/ibm/console`
2. Enter the administrator account and password.
   - For example, default user name: `fimadmin`
   - Password as specified when the console was installed.

   The console Welcome panel opens. The navigation pane, on the left side of the window, lists an entry for IBM Tivoli Federated Identity Manager.
3. Click the icon next to IBM Tivoli Federated Identity Manager.
   - The Welcome portlet, in the right panel, lists the installed Integrated Solutions Console plug-ins.
   - The About portlet, also in the right panel, describes the version information for the Integrated Solutions Console.
4. To log out, click **Logout** in the upper right corner.

Task grouping overview

The IBM Tivoli Federated Identity Manager groups administration tasks into five different categories for easier access and tracking.

The console organizes administration tasks into categories (task groupings). Each category is listed in the left navigation pane.

To view the categories:

1. Log on to the Integrated Solutions Console as described in "Logging in and logging out" on page 1.

   The left pane shows categories and tasks for WebSphere Application Server and IBM Tivoli Federated Identity Manager. The IBM Tivoli Federated Identity Manager tasks are grouped together under the **IBM Tivoli Federated Identity Manager** section.
2. Expand the IBM Tivoli Federated Identity Manager section by clicking the + icon next to each task.

The IBM Tivoli Federated Identity Manager administration tasks are grouped in the following sections:

- **Getting Started**
- **Configure Federated Single Sign-on**

  `Administration tasks for managing single sign-on federations`
  Administration tasks for single sign-on federations. Entry point for

managing, creating, modifying, and deleting federations. Entry point for exporting federation configuration. You can also add partners to existing federations through this entry point.

Partners
Administration tasks for managing partners in single sign-on federations. Entry point for adding partners, modifying partner configuration, and deleting partners.

- **Configure Trust Service**

Module Types
Administration tasks for managing each type of security token module. Entry point for creating module types, modifying module type properties, and deleting module types.

These entry points are used primarily for configuring Web services security management deployments. For federated single sign-on, the default module types are used, and typically are not modified or deleted. However, you can create new module types if necessary, and token module types can be managed through the Federation wizard and the Partner wizard.

Module Instances
Administration tasks for managing instances of each type of security token module. Entry point for creating module instances, modifying module instances properties, and deleting module instances.

These entry points are used primarily for configuring Web services security management deployments. For federated single sign-on, token module instances are managed through the Federation wizard and the Partner wizard.

Trust Service Chains
Administration tasks for managing chains of security token module instances. Entry point for creating trust service chains, modifying trust service chain contents, and deleting trust service chains.

These entry points are used primarily for configuring Web services security management deployments. For federated single sign-on, trust service chains are managed through the Federation wizard and the Partner wizard.

- **Configure Key Service**

Keystores
Administration tasks for managing keys and certificates that are used to sign assertions, validate signatures, and encrypt data. Entry point for importing keystores, deleting keystores, importing and exporting keys, enabling or disabling keys, and deleting keys.

Hardware Cryptographic Device
Administration tasks for managing configuration of hardware cryptographic devices. Entry point for specifying whether to use a hardware cryptographic device, and for specifying the location of the configuration file for the device.

- **Domain Management**

Runtime Node Management
Administration tasks for deploying the runtime node after updating it through a fix pack or language pack, and customizing the runtime service of IBM Tivoli Federated Identity Manager, publishing custom event pages and STS modules, and reloading configurations.

Import and Export Configuration
Administration tasks for importing or exporting the configuration of the IBM Tivoli Federated Identity Manager domain. Entry point for exporting the current configuration to a new configuration archive or importing an archive to a new location.

Point of Contact
Administration tasks for managing custom point of contact servers. Entry point for creating, activating, or deleting custom point of contact servers and for selecting endpoint and authentication settings.

Event Pages
Administration tasks for customizing message and error pages or page locales.

Alias Service Settings
Administration tasks for setting the database to use for storing aliases and attributes.

Auditing
Administration tasks for auditing program activities. Entry point to enable audit logging, specify the location for the log, and specify the size and number of logs before the current log file rolls over to a new log file.

Web Server Plugin Configuration
Administration tasks for configuring IBM Tivoli Federated Identity Manager plug-ins for Web servers. Tasks include enabling plug-ins, specifying access to LTPA cookies, logging, configuring applications, and exporting Web server configuration files.

Reports
Administration tasks for configuring, running, and viewing audit reports. Includes tasks for configuring report settings.

Domains
Administration tasks for creating, deleting, and activating domains. Includes access points for showing domain information and modifying domain properties.

Task portlets

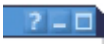
You can view tasks easily in a portlet because tasks are arranged in a more organized manner.

The console opens a separate portlet for each administration task. A portlet is represented as a panel. A panel is an individual section within a console window.

The navigation pane is on the left side and one or more portlet panels are on the right side. The content of the portlet panel is determined by the task that you have selected in the navigation pane.

For example, when you log on to the console and select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Import and Export Configuration**, the Export Configuration portlet and the Import Configuration portlet opens on the right panel.

Portlets act similarly to windows. For example, you can minimize or maximize portlets by clicking their corresponding icons.

The maximize icon  , and the shared screen icon  are located in the group of icons in the upper right corner  of the portlet.

Task wizards

The console provides task *wizards* to guide you through the more complex tasks.

Each wizard opens a series of panels that prompt you for the necessary configuration input. The choices you make at each panel can determine which additional panels open.

The wizards have a common format for presentation in the right panel of the console. Each wizard has a left panel that opens the tasks in the order in which they are completed. The panel includes an arrow icon that indicates the current task. The arrow moves through the task list as each task is completed.

Wizards have a standard set of action buttons at the bottom of each panel:

- Next

  Continue to the next panel within the wizard.
- Back

  Return to the previous wizard panel. This action is useful when you must adjust a prior configuration setting.
- Cancel

  Cancel the entire administration task. When you select Cancel, the wizard exits and all configuration settings that you have entered are discarded.
- Finish

  Commit the configuration settings that have been entered, and exit the wizard. This button is enabled only on the last panel of the wizard.

Help

Learn where you can get help for administration task panels and for console functions. Also, find out how to access the information center for thIBM Tivoli Federated Identity Manager from the console.

The console provides several ways to obtain help:

- "Online help for administration task panels" on page 6

  The console presents one or more configuration panels for each administration task. Each panel has an associated help file.
- "Online help for console functions" on page 6

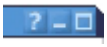
  The Integrated Solutions Console has a help system that describes the use of the console to perform functions that are common to all administration task plug-ins. The help system is not specific to a product that is using the Integrated Solutions Console. Instead, this help section describes how to best use the console to administer all plug-ins.
- "Accessing the Tivoli Federated Identity Manager information center" on page 6

  Within the Integrated Solutions Console help system, you can access an entry point for IBM Tivoli Federated Identity Manager. At this entry point, you can click a link to access the IBM Tivoli Federated Identity Manager information center on the IBM website.

Online help for administration task panels

The online help panels are independent panels that apply to one task only. They provide context-sensitive help for the current task and do not link to other tasks.

Each administration task panel has an online help panel.

To access the online help for a panel, click the question mark  in the upper right corner  of the blue bar. An online help panel for working with this portlet shows.

Online help for console functions

The Integrated Solutions Console provides context- sensitive help for the Integrated Solutions Console features.

The Integrated Solutions Console window always provides a Help button in the upper right corner of the full window. This button shows regardless of which plug-in module is being used to perform administration tasks.

When you click **Help**, the help information for Integrated Solutions Console opens. This information describes navigation of the console and use of the base administration pages. The help system contains many pages.

You can consult the help pages to determine how to best use Integrated Solutions Console features such as filtering and ordering of table entries. The filtering and ordering feature can be useful when you have many table entries. For example, for IBM Tivoli Federated Identity Manager you might have many keys in the Key service table, or many partners in the Federation partners table.

Accessing the Tivoli Federated Identity Manager information center

The IBM Tivoli Federated Identity Manager information center is available on the Internet.

Procedure

1. From any console panel, click **Help** in the upper right corner.
   The Console Basics Help panel opens.
2. The Integrated Solutions Console Basics Help panel contains a navigation hierarchy in the left panel. Select **IBM Tivoli Federated Identity Manager**.
   The IBM Tivoli Federated Identity Manager Welcome panel opens.
3. The Welcome panel contains a link to the IBM Tivoli Federated Identity Manager information center on the IBM website. Click this link to access the entire IBM Tivoli Federated Identity Manager information set.

Chapter 2. Managing administrators

You can add, delete, modify IBM Tivoli Federated Identity Manager administrator names and roles. One administrator user is configured at the time IBM Tivoli Federated Identity Manager is installed.

About this task

The name for this administrator user depends on the version of WebSphere Application Server used with the installation.

Version of WebSphere Application Server	Default administrator name
Existing version	The administrator name and password that were configured on WebSphere Application Server when IBM Tivoli Federated Identity Manager was installed.
Embedded version	`fimadmin` is the default administrator name. The password is assigned during installation.

You might want to modify this administrator, add one or more administrators, or delete an existing administrator.

You can also manage the roles of administrator users and organize and manage administrator users in groups. For help with these topics, see the online help in the console.

Modifying administrator properties

Use the **Users and Groups** task in the console to modify the email address, user name, password, or group properties for an existing administrator in your WebSphere Application Server and Tivoli Federated Identity Manager environment.

Procedure

1. Log on to the console.
2. Click **Users and Groups** > **Manage Users**.
3. Locate the existing users by selecting the search criteria.
4. Click **Search**.
5. Click the user ID of the user you want to modify.
6. Complete the fields. (Optional) Click **Groups** to modify the group properties.
7. Click **Apply**.
8. Click **OK**.

Adding an administrator

Use the **Users and Groups** task in the console to add an administrator.

About this task

Attention: Only one administrator must use the console at a time to avoid conflicting changes in the console.

Procedure

1. Log on to the console.
2. Click **Users and Groups** > **Manage Users**.
3. Locate the existing users by selecting the search criteria.
4. Click **Search**.
5. Click **Create**.
6. Complete the fields as required. (Optional) Click **Group Membership** to configure group properties for the administrator.
7. Click **Create**. A message shows when the user is created.
8. Click **Create Like** to create another user by using properties from the one you created, or click **Close** to complete the task.
9. Add the appropriate permissions for the administrator you created:
   a. Click **Users and Groups** > **Administrative User Roles**.
   b. In the Administrative User Roles table, click **Add**.
   c. Type the administrator name in the **User** field and select a role (either **Administrator** or **fimadmin**) from the list.
   d. Click **OK**.
   e. Click **Save** to save directly to the master configuration. Changes take place immediately.

Deleting an administrator

Use the **Users and Groups** task in the console to delete an administrator.

About this task

Attention: If you have only one administrator account, do not delete it.

Procedure

1. Log on to the console.
2. Click **Users and Groups** > **Manage Users**.
3. Locate the existing users by selecting the search criteria.
4. Click **Search**.
5. Select the check box next to the user ID for the user you want to delete.
6. Click **Delete**.
7. You are prompted to verify that you want to delete the selected user. Click **OK** or **Cancel**.

Chapter 3. Modifying LTPA key and timeout settings

Review the Lightweight Third Party Authentication (LTPA) settings on your WebSphere Application Server after you have installed IBM 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. The keys are 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, the LTPA keys must be on your WebSphere Application Server point of contact server and on your target application server. Examples of separate target applications are, separate WebSphere Application Server or an IHS, or IIS server. If you automatically generate keys, keep the keys on the application server in sync with the keys that are generated on your WebSphere Application Server point of contact server.

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.

Procedure

1. Log on to the console.
2. Click **Security** > **Secure administration, applications, and infrastructure**.
   The Secure administration, applications, and infrastructure panel opens.
3. On the right, click **Authentication mechanisms and expiration**. The Configuration tab shows.
   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**. Change your environment appropriately, 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.

Chapter 4. Modifying identity mapping configurations

When you configured your federation and added your partners, you selected the appropriate identity mapping method for the federation or to be used with each partner. Your identity mapping configuration is listed in your Federation properties. If you configured specific identity mapping settings for your partner, that configuration is listed in your Partner properties.

You have two options for modifying these configurations:

- "Changing the identity mapping module instance"
- "Modifying identity mapping properties" on page 12

Changing the identity mapping module instance

You can change whether to use an XSLT and JavaScript transformation file or a custom mapping module to process the identity mapping for an existing federation or an existing partner in the federation.

Before you begin

Attention: Before continuing with this procedure, make sure that you are familiar with the identity mapping options. You must also complete all of the required configuration for those options. See identity mapping topics and the token and content format topics in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

About this task

Note: Identity mapping applies to partners of all protocols except for OAuth 1.0 and 2.0.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on**.
3. Click **Federations** or **Partners**. The panel shows a list of configured federations or partners.
4. Select a federation or partner.
5. Click **Properties** to view the properties for an existing federation or partner.
6. Locate the **Identity Mapping Properties** section and click **Change Identity Mapping Module Instance**. The Identity Mapping Options panel opens.
7. Select the identity mapping option you want to use.

   XSLT and JavaScript transformation file
   If you choose this option, you must provide a path and file name for the XSLT and JavaScript transformation file.

   Use Tivoli Directory Integrator for identity mapping
   When you choose this option, the Tivoli Directory Integrator module is placed in the chain as the mapping module. If you want to modify the default configuration properties for the module, select **Modify Current Properties**.

Custom mapping module
If you choose this option, you must select a module instance from the list that shows.

8. Click **OK** when you have finished.

Modifying identity mapping properties

After the initial configuration of your federation and partner, you can modify the identity mapping properties that are defined for a federation or the partner.

Before you begin

XSL transformation file
If you selected an XSL transformation file, you can specify a new file to use.

Tivoli Directory Integrator module
If you selected Tivoli Directory Integrator module, you can modify the configuration properties.

Custom mapping module
If you selected a custom mapping module, you can modify the module properties.

About this task

Note: Identity mapping applies to partners of all protocols except for OAuth 1.0 and 2.0.

To modify identity mapping properties:

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on**.
3. Click **Federations** or **Partners**.
4. The panel shows a list of configured federations or partners. Select a federation or partner.
5. Click **Properties** to view the properties for an existing federation or partner.
6. Locate the **Identity Mapping Properties** section and click **Modify Current Properties**.
   - When the federation or partner uses an XSL transformation file, the Identity Mapping Rule portlet opens. You can modify or delete the identity mapping rule in the Identity Mapping Rule portlet.

     Attention: Ensure that you can access the mapping file from the computer on which you are using a browser to view the console. For example, put the mapping file on the computer where you are viewing the console. You can also map a drive on the computer where you are viewing the console to a drive where the identity mapping file is located.
   - When the federation or partner uses the Tivoli Directory Integrator module, the Identity Mapping Properties panel opens.
   - When the federation or partner uses a custom mapping module, a portlet opens where you can view or modify the custom mapping instance properties.

7. The action you take depends on whether you are modifying a transformation file or a custom module:
   - If you are modifying the settings for an XSL transformation file, click **Modify Rule** or **Delete Rule**. Then, take the appropriate action as described on the portlet.
   - If you are modifying the Tivoli Directory Integrator module, make the appropriate changes on the Identity Mapping Properties panel.
   - If you are modifying custom module properties and the module has configuration settings that can be changed, make the appropriate changes.
8. Click **OK** when you have finished.

Chapter 5. Managing federations

You can use the console to manage federations.

For information about establishing a federation, including creating a federation and adding a partner to an existing federation, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Management tasks include:

- "Modifying federation properties"
- "Viewing your partners in a federation"
- "Exporting federation properties" on page 16
- "Deleting a federation" on page 16

Modifying federation properties

Use the Federations properties selection to view the details about an existing federation or to modify an existing federation.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel shows a list of configured federations.
3. Select a federation.
4. Click **Properties** to view the properties for an existing federation.
5. Select the properties to modify. Federation properties are described in the online help.
6. When you have finished modifying the properties, click **OK** to close the Federation Properties panel.

Viewing your partners in a federation

The View partners option on the Federation panel shows a list of partners that are configured for the specified federation.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel shows a list of configured federations.
3. Select a federation.
4. Click **View Partners** to view the partners for the selected federation.

What to do next

You can perform the following management tasks on the partners listed:

- "Enabling or disabling a partner" on page 17
- "Modifying partner properties" on page 17

- "Deleting a partner" on page 19

Information about adding a partner is available in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Exporting federation properties

When you want to join a federation hosted by another business partner, 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 on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel opens.
3. Select a federation from the table.
4. Click **Export**. The browser shows a message window that prompts you to save the file containing the exported data.
5. Click **OK**. The browser download window prompts for a location to save the file.
6. Select a directory and metadata file. 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

   Note: Place the metadata file in an easily accessible location. You must provide the file to your partner, when your partner wants to import configuration information for the federation.
7. Click **Save**.

Deleting a federation

Use the Integrated Solutions Console to delete the federation that you no longer need.

About this task

Attention: When you delete a federation, all of its partners are also deleted.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel shows a list of configured federations.
3. Select a federation.
4. Click **Delete** to delete the federation. A message box prompts you to confirm the deletion of the federation.
5. Click **OK** on the message box. The federation is deleted.

Chapter 6. Managing federation partners

You can use the console to manage partners.

For information about adding a partner to an existing federation, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Management tasks include:

- "Enabling or disabling a partner"
- "Modifying partner properties"
- "Deleting a partner" on page 19

Enabling or disabling a partner

You must enable a partner to activate federated single sign-on functions. If you do not want a partner to have any access to a federation, disable it.

About this task

When you add a partner to a federation, the partner is disabled by default as a security precaution. You must enable the partner in order for it to establish a connection with the federation.

When necessary, you can disable partner access without removing the partner configuration. You must disable a partner temporarily, for example, while replacing an expired key or certificate.

To enable or disable a partner:

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federation Partners panel opens.
3. Select the partner from the partner table. When you select a partner, the **Enable/Disable** button is activated.
   - When the partner is disabled, the **Enable/Disable** button is redrawn. Click **Enable** to enable the partner.
   - When the partner is enabled, the **Enable/Disable** button is redrawn. Click **Disable** to disable the partner.

Modifying partner properties

You can modify partner properties at any time after you have added the partner to your federation.

About this task

You can modify partner properties at any time after you have added the partner to your federation. For each partner, you can access a Partner Properties panel that shows the partner configuration.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federation Partners panel opens.
3. Select the partner from the partner table.
4. Click **Properties**. The Partner Properties panel opens.
5. Select the properties to modify. See the online help for descriptions of the properties.
6. When you have finished modifying properties, click **OK** to close the Partner Properties panel.

Using a keystore for multiple SOAP endpoint server certificate validation

Use a keystore for multiple SOAP endpoint server certificate validation.

About this task

If two or more SOAP endpoints use HTTPS, you must use a keystore to access the server validation certificates. The keystore must contain the multiple server validation certificates used to validate the endpoints. Edit the partner response file to specify the keystore to use in validating the SOAP endpoints.

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, or Solaris

   ```
   wsadmin.sh
   ```

   Note: For more information about the options that can be specified when you run the wsadmin tool, see the WebSphere Application Server Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.
3. Save the partner configuration properties into a response file with the following command: `$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName` *name* `-federationName` *name* `saml20sp -partnerName` *name* `ibmidentitydemo -fileId /tmp/partner.out}`
4. Locate the `ServerCertKeyId` property in the response file and identify the value to be deleted:

   ```
   <void method="put">
      <string>ServerCertKeyId</string>
      <object class="java.util.ArrayList">
       <void method="add">
        <string>DefaultTrustedKeyStore_certname</string>
       </void>
      </object>
     </void>
   ```

   Note: In this example the value of the `ServerCertKeyId` property is `DefaultTrustedKeyStore_certname`.

5. Delete the value of the ServerCertKeyId property in the response file and insert the ServerCertKeystoreId property:

```
<void method="put">
   <string>ServerCertKeyId</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string></string>     <------ This value was removed
    </void>
   </object>
  </void>

<!-- This whole property was added -->
  <void method="put">
   <string>ServerCertKeystoreId</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>DefaultTrustedKeyStore</string>     <------ This value was added
    </void>
   </object>
  </void>
```

6. Use the following commands to update the configuration in IBM Tivoli Federated Identity Manager:

```
$AdminTask manageItfimPartner {-operation modify
-fimDomainName name -federationName name saml20sp
-partnerName name ibmidentitydemo -fileId /tmp/partner.out}
```

7. Use the following command to reload the IBM Tivoli Federated Identity Manager configurations:

```
$AdminTask reloadItfimRuntime {-fimDomainName name}
```

   You can also use the console to reload the IBM Tivoli Federated Identity Manager.

Deleting a partner

Delete the partner configuration to remove partner access and connections.

About this task

When you no longer want to maintain a trust relationship with a partner, you can delete the partner configuration from your federation configuration.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federation Partners panel opens.
3. Select a partner from the partner table.
4. Click **Delete**. A message box shows to prompt you to confirm the deletion of the partner.
5. Click **OK**.

Chapter 7. Managing modules

If you choose to use a custom mapping module in your federation, you must have added it as a module type and as a module instance to the IBM Tivoli Federated Identity Manager environment.

For information about creating custom mapping modules, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Use the console to perform the following module management tasks:

Viewing module type properties

You can view properties for custom modules that you have created.

Procedure

1. Select **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Module Types**. The Module Types panel opens.
2. Select a module type.
3. Click **Properties** to view the properties of an existing token module type. The Module Properties panel opens.
4. Click **OK** when you are done.

Modifying module types

You can modify custom modules that you have created by creating and publishing new ones.

About this task

Attention: Do not modify the default module types that are installed by default with IBM Tivoli Federated Identity Manager. Use this task only to modify module types that you have added.

To modify a module type, you must modify it outside of the IBM Tivoli Federated Identity Manager console, and then republish it.

Procedure

1. Modify the module and save it to your /plugin directory.
2. Log on to the console.
3. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Runtime Node Management**. The Runtime Node Management panel opens.
4. Click **Publish Plug-ins**. The new module becomes available and shows in the Module Types list.

Deleting a module type

Delete a custom module type only when it is no longer needed.

About this task

Attention: Do not delete the default module types that are installed by default with IBM Tivoli Federated Identity Manager. Use this task only to delete module types that you have added.

To delete a module type, you must remove it from the plug-in directory and then republish your remaining plug-ins to reflect the deletion.

Procedure

1. Remove the module from the `/plugins` directory.
2. Log on to the console.
3. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
4. Click **Publish Plug-ins**.

What to do next

After publishing the plug-ins, the deleted module must be removed. Deleted mapping modules are removed from the Module Types list.

Creating a module instance

You must create an instance of a module type to use the module type with the trust service. The management console provides a wizard to guide you through this task.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Module Instances**. The Module Instances panel shows the module instances that are created by default. It also shows any module instances that you have added.
2. Click **Create**. The Token Type panel shows 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.
4. Click **Next**. The Module Instances wizard opens the Module Instances Name panel.
5. Enter values for the requested properties, and click **Finish**.

Modifying module instance properties

You can modify properties for each module instance. For example, when you add more than one instance of a module type, you must update the name or description for an existing module instance.

About this task

Attention: Do not change the properties of a default IBM Tivoli Federated Identity Manager module instance. Change the properties only of custom modules you have added.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Module Instances**. The Module Instances panel opens.
2. Select a module instance.
3. Click **Properties** to view or modify properties for an existing token module instance. The Module Instance Properties panel opens.
4. Select the properties to modify for the selected module instance. See the online help for property descriptions.
5. Click **Apply** to save changes without leaving the Module Instance Properties panel. Click **OK** to save the changes and exit from the Module Instance Properties panel.

Deleting a module instance

Use the Integrated Solutions Console to delete a module instance that is no longer needed.

About this task

Attention: Do not delete the default module instances that are installed by default with IBM Tivoli Federated Identity Manager. Use this task only to delete module instances that you have added.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Module Instances**. The Module Instances panel opens.
2. Select a module instance.
3. Click **Delete**. A message box prompts you to confirm whether to delete the specified module instance.
4. Click **OK** to delete the module instance or click **Cancel** to exit from the window without deleting the module instance.

Creating a trust service chain

Use the management console to configure custom trust service chains.

About this task

To create a trust service chain, you must assign a module instance to each link in the chain, and specify a mode for each instance. The Trust Service Chain wizard guides you through the tasks.

Before you create a chain, you must understand the concepts of module types, module instances, module modes, and trust service chains.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Trust Service Chains**.

   The Trust Service Chains panel opens.
2. Click **Create**. The Trust Service Chain wizard starts. The Introduction panel opens.
3. Read the introduction.
4. Click **Next** The Chain Mapping Identification panel opens.
5. Specify the requested properties.

   See "Chain mapping identification properties" on page 28.
6. Click **Next**. The Chain Mapping Lookup panel opens.
7. Specify values for each of the required properties.

   For more information about the properties, see "Chain mapping lookup properties" on page 28.
8. When finished, click **Next**. The Chain Identification panel opens.
9. Specify values for each of the required properties.

   See "Chain Identification properties" on page 31.
10. When finished, click **Next**. The Chain Assembly panel opens.
11. Assemble the chain:
    a. Select a module instance from the **Module Instance** menu.
    b. Select a mode for the instance, from the **Mode** menu.
    c. Click **Add Selected Module Instance to Chain**.

    The Trust Service Chain Modules table is updated, and now opens the module instance that you have added.
12. Repeat the previous step for each module instance to add to the chain.

    The Chain Assembly panel enables you to reorder the chain. Notice that each entry you have created has an integer value next to it. The integers reflect the order in which you added the instance to the chain. If you want to reorder the instances:
    a. Change the integer value for each instance to the correct value.
    b. Click **Reorder**.
13. When the order of the instances in the trust service chain is correct, click **Next**.

    When you specify a custom trust service chain that does not match the typical chain pattern, the panel opens a warning. The warning states that most chains consist of either a single instance, in issue or validate mode, or consist of a sequence of validate > map > issue.

    When you choose to create a chain that uses an atypical chain pattern, you can click **Continue** within the warning box.
14. The next panel prompts for configuration information for the first module instance in the chain. The contents of the panel are determined by the module type. The wizard presents a panel for each module instance that requires configuration. Some modules do not require configuration.

    For information about the properties for each module type, see "Supported module types" on page 31.
15. When all module instances have been configured, click **Next**. The Summary panel opens.

16. Review the summary of the trust service chain. If you want to change any properties, return to the appropriate panel.
17. When you are finished with the configuration, click **Finish**.

Results

The trust service chain is created.

Creating trust chain like an existing chain

You can use an existing trust chain configuration as a template for creating a trust chain.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Trust Service Chains**.

   The Trust Service Chains panel opens.
2. Click **Create Like**. The Trust Service Chain wizard starts. The Introduction panel opens.
3. Read the information.
4. Click **Next**. The Chain Mapping Identification panel opens.

   See “Chain mapping identification properties” on page 28.
5. Specify the requested properties.
6. Click **Next**. The Chain Mapping Lookup panel opens.

   For more information about the properties, see “Chain mapping lookup properties” on page 28
7. Specify values for each of the required properties.
8. When finished, click **Next**. The Chain Identification panel opens.

   See “Chain Identification properties” on page 31.
9. Specify values for each of the required properties.
10. When finished, click **Next**. The Chain Assembly panel opens.

    The existing trust chain module configuration shows.
11. Apply the task that is appropriate to your deployment:
    - When you chose to clone an existing chain, the panel opens the modules in the existing chain. Confirm that the modules are correct for your deployment, and click **Next** to continue.
    - When you chose to reuse an existing chain, confirm that the modules are correct for your deployment. Next, specify whether to share federation properties.

      Select the federation properties check box if you want this trust chain mapping to share the federation properties of the trust chain mapping that you are reusing. This trust chain mapping has a reference to the same self configuration as the trust chain mapping you are reusing.

      When you are prompted for module instance settings, you must provide only the partner properties. If cleared, you must provide the self and partner properties for the federation.
12. The next panel prompts for configuration information for the first module instance in the chain. The contents of the panel are determined by the module type. The wizard shows a panel for each module instance that requires configuration. Some modules do not require configuration.

For information about the properties for each module type, see "Supported module types" on page 31.

13. When all module instances have been configured, click **Next**. The Summary panel opens.
14. Review the summary of the trust service chain. If you want to change any properties, return to the appropriate panel. When you are finished with the configuration, click **Finish**.
15.

Results

The trust service chain is created.

Modifying trust service chain properties

You can use the management console to view or modify lookup properties for existing trust service chains.

About this task

You can use this console entry point to modify custom trust service chains. Use this entry point to modify the chain lookup properties.

The trust service chain management function of the management console does not show chains that are automatically generated. The automatically generated chains include chains generated for federated single sign-on.

Note: Do *not* modify the built-in SSO trust chains. See the topic on Complex Federation Identity and Attribute Mapping for Tivoli Federated Identity Manager from the IBM community blogs to know why modifying the built-in SSO trust chains is not good architectural approach.

To modify chains for federated single sign-on, access the federation properties panel for the single sign-on federation through **Configure Federated Single Sign-on → Federations**.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Trust Service Chains**. The Trust Service Chains panel opens.
2. Select a trust service chain.
3. Click **Properties** The Trust Service Chain Mapping Properties panel opens.
4. You can view or modify the chain lookup properties for the trust service chain. For more information about each property, see:
   - "Chain mapping identification properties" on page 28
   - "Chain mapping lookup properties" on page 28
   - "Chain Identification properties" on page 31
5. When you are finished viewing or modifying properties, click **Apply** to save modifications without exiting the properties panel.
6. Click **OK** to exit the panel.

Modifying chain module properties

Use the management console to modify properties for module instances that have been added to a trust service chain.

About this task

You can use the console to modify custom trust service chains. The trust service chain management function of the management console does not show chains that are automatically generated. The automatically generated chains include chains generated for single sign-on federations.

Note: Do *not* modify the built-in SSO trust chains. See the topic on Complex Federation Identity and Attribute Mapping for Tivoli Federated Identity Manager from the IBM community blogs to know why modifying the built-in SSO trust chains is not a good architectural approach.

To modify chains for federated single sign-on, access the federation properties panel for the single sign-on federation through **Configure Federated Single Sign-on → Federations**.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Trust Service Chains**. The Trust Service Chains panel opens.
2. Select a trust service chain and click **Modify Chain**.
   The Modify Module Chain panel opens.
3. Modify the chain properties as required.
   When you finish modifying a property you can click **Apply** to save the change without exiting the portlet. You can make the following changes:
   - Add a module instance to the chain
     a. Select a module instance from the **Module Instance** menu.
     b. Select a mode for the instance from the **Mode** menu.
     c. Click **Add Selected Module Instance to Chain**.
   - Delete a module instance from the chain
     a. Select a module instance from the Trust Service Chain table.
     b. Click **Delete**.
   - Reorder the module instances within the chain
     a. Select a module instance from the Trust Service Chain table.
     b. Change the integer value in the Order column to the required value.
     c. Repeat the previous two steps for each module instance that requires reordering.
     d. When finished, click **Reorder**.
   - Modify the configuration properties for each module within the chain.
     a. Select a chain mapping.
     b. Click **Properties**.
     c. Select a module instance from the Trust Service Chain table.
     d. Click **Properties**.
     e. Modify the properties as required. The properties are specific to the module type and module mode. For more information, see "Supported module types" on page 31.

4. When you have finished modifying the properties, click **OK** to save changes and exit the portlet.

Deleting a trust service chain

Use the management console to delete a trust service chain that is no longer needed.

Procedure

1. Click **IBM Tivoli Federated Identity Manager → Configure Trust Service → Trust Service Chains**. The Trust Service Chains panel opens.
2. Select a trust service chain.
3. Click **Delete**. A message box shows and prompts you to confirm whether you want to delete the specified trust service chain.
4. Click **OK** to delete the trust service chain or click **Cancel** to exit from the window without deleting the chain.

Chain mapping identification properties

Each chain must have a unique name. You can optionally specify a description string for the chain.

Chain Name
(Required) Choose an arbitrary string name. The console does not limit the string name other than the limitation imposed by the Java™ data type. Consider keeping the string name under 40 characters, for ease of use when viewing the name on the IBM Tivoli Federated Identity Manager console.

Description
(Optional) The console does not limit the string size other than the limitation that is imposed by the Java data type.

Create a dynamic chain
Select this check box if you want to create a dynamic chain. A dynamic chain, also called an application chain, is separately configured and is executed after a protocol chain is executed. The benefit of this is that a single application chain might be dynamically attached to more than one protocol chain in a many-to-one relationship. The protocol chain must be configured separately from the dynamic chain.

Chain mapping lookup properties

This topic describes the properties that the trust service uses to map an incoming request to the correct trust service chains.

The Chain Mapping Lookup panel shows a set of properties that the IBM Tivoli Federated Identity Manager trust service consults when processing an incoming request. Doing so determines which trust chain to use. When matching data is found, the trust chain starts.

Note: The incoming request is part of a WS-Trust standard for conveying token data structure.

Request type properties

Request Type
The type of request to associate with this chain. The request is one of the types supported by the WS-Trust specification:

Issue
Issue a new token, based on information obtained from the request.

Renew
Renew a token that has expired.

Validate
Validate the specified security token and return the requested result.

Cancel
Cancels a previously issued token so that it is no longer used.

Key Exchange
Transfer of a new key for use by the receiver of the request.

Other
A custom request type.

Request Type URI
A Uniform Resource Indicator for each request type. The fields are read-only, except for the Other type. For the Other type, enter the URI for your custom request type.

Lookup type properties

Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)
Specifies that the trust service uses the values in the request for AppliesTo, Issuer, and TokenType as input to determine which trust service module chain to call.

Select this option to show the data entry fields for AppliesTo, Issuer, and TokenType.

Use XPath to Define Custom Lookup Rule
Specifies that a custom lookup rule is to be defined. Specifies to use XML Path Language (XPath) rule to use to find the location of the custom rule.

Select this option to show a text box where you can specify the XPath rule.

XPath
A text field that can be used to define a custom lookup rule. Use XML Path Language to define the rule. During chain configuration, the console shows the text field only when the **Use XPath to Define Custom Lookup Rule** option has been selected.

AppliesTo
Defines the scope of the token.

Address
The address of the service that is being requested. For example:

```
http://sp.example.com:9080/TrustServer/SecurityTokenService
```

When specifying scope for single sign-on protocols, this URI is sufficient information for identifying the issuer.

Service Name
A qualified name (QName) that includes a namespace URI and a local part for the Web service.

Note: A QName is a term defined in XML specifications. A QName consists of a namespace URI plus a local part.

For example:

```
http://sp.example.com:SecurityTokenService
```

Note: The Service Name data entry field provides a colon (:) to separate the namespace URI from the local part.

This field is typically not used for single sign-on protocols, but can be used for Web services security management.

Port Type

Web services operations are grouped by port type. Use this field when there are more than one Web services port types to specify.

For example, a stock market data service might provide both a stock quoting service and a stock price history service. Use this field to specify the needed Web service.

```
http://sp.example.com:RequestSecurityTokenPort
```

The port type is also a QName.

Note: The Port Type data entry field provides a colon (:) to separate the namespace URI from the local part.

This field is typically not used for single sign-on protocols, but can be used for Web services security management.

Issuer

Address

The URL for the company or enterprise that issued the token. For example:

```
example.com
```

When specifying scope for single sign-on protocols, this provider ID is sufficient information for identifying the issuer.

Service Name

A qualified name (QName) that includes a namespace URI and a local part for the Web service.

Note: A QName is a term defined in XML specifications. A QName consists of a namespace URI plus a local part.

For example:

```
http://idp.example.com:myWebService
```

Note: The Service Name data entry field provides a colon (:) to separate the namespace URI from the local part.

This field is typically not used for single sign-on protocols, but can be used for Web services security management.

Port Type

Web services operations are grouped by port type. Use this field when there is more than one Web service port type to specify.

For example, a stock market data service might provide both a stock quoting service and a stock price history service. Use this field to specify the needed Web service.

`http://idp.example.com:getQuotePortType`

The port type is also a QName.

Note: The Port Type data entry field provides a colon (:) to separate the namespace URI from the local part.

This field is typically not used for single sign-on protocols, but can be used for Web services security management.

Token type property

Token Type
Select the token type from the menu. See the online help for information.

Chain Identification properties

You can use the management console to modify properties for the chain identification.

Chain Name
(Required) You can choose an arbitrary string name. The console does not limit the string name other than the limitation imposed by the Java data type. Consider keeping the string name under 40 characters, for ease of use when viewing the name on the IBM Tivoli Federated Identity Manager console.

Description
(Optional) The console does not limit the string size other than the limitation that is imposed by the Java data type.

Initialize the chain upon startup of the Runtime
Select this check box when you want the chain initialized automatically when the IBM Tivoli Federated Identity Manager runtime starts.

When you use the **Create Like** trust chain wizard to build a chain, you can choose how to use the configuration of an existing chain:

Reuse existing chain
You can reconfigure the properties for the chain that you selected before starting the wizard.

Clone existing chain
You can create an identical copy of an existing chain. You must supply a name for the new chain in the **Chain Name** field.

Supported module types

IBM Tivoli Federated Identity Manager supports a number of module types.

Note: Liberty protocol is being deprecated in the Tivoli Federated Identity Manager 6.2.2 release.

Attention: The following topics provide configuration information for existing modules that are to be added to a custom trust chain. For more information about developing modules and building custom trust chains, see the *IBM Tivoli Federated Identity Manager Configuration Guide* and IBM developerWorks® articles. You are also expected to be familiar with any applicable security standards for these modules.

Each module that is added to a trust service chain must be configured. Configuration properties are specific to both the module type and module mode within the chain. Some combinations of module type and mode do not require configuration.

Authorization module

The Authorization module is called AuthorizationSTSModule.

This module interacts with the authorization engine, such as Tivoli Access Manager for e-business to run an authorization check for the supplied user identity.

Note: This module has been deprecated and is included only for backwards compatibility. Use the Tivoli Access Manager for e-business authorization module instead. See "Tivoli Access Manager for e-business authorization module" on page 62.

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Other

Configuration properties

Web Service protected object name
The configured protected object name of the Web Service.

The authorization module combines the configured protected object name of the web service with the port type and operation contained in the AppliesTo. The combination forms the protected object name for the web service operation being started.

For example, if the configured protected object name of the web service is `/itfim-wssm/wssm-default/EchoWSDL/EchoService` and the AppliesTo includes a port type of `EchoService` and operation of `echo`, then the protected object name of the web service operation is:

```
/itfim-wssm/wssm-default/EchoWSDL/EchoService/EchoService/echo
```

This protected object name must match the object name configured in the Tivoli Access Manager for e-business protected object space.

Custom runtime properties
The `AuthorizationSTSModule` connects to Tivoli Access Manager for e-business to make authorization decisions. For optimum performance, it maintains a pool of connections. When an error happens, the connection used is marked for removal from the pool. Two custom runtime properties can control the number of connections created and maintained by the pool:

- `authorizationsts.initial.num.context`

  Specifies the initial amount of context objects to be created at startup.
- `authorizationsts.max.num.context`

  Specifies the maximum amount of context objects to be created throughout.

See "Creating a custom property" on page 158 for information on how to define the custom properties.

Default mapping module

The Default mapping module is called XSLTransformationModule.

The default map token consists of an eXtensible Stylesheet Language (XSL) file that specifies an identity mapping rule. The module calls an XSL parser to read identity mapping rules to generate a Secure Token Service Universal User XML document. The generated Secure Token Service Universal User XML document contains the user identity information.

Deployment scenarios for this module type

- Single sign-on federations
- Web services security management
- Custom trust chains

Supported modes

- Map

Configuration properties
The Trust Service Chain wizard prompts for the name of the file that contains the identity mapping rule.

XSLT file Containing Identity Mapping Rule
The name of the file that contains the identity mapping rule. The file must be written and complete before you can configure it into the chain. For example:

/tmp/ip_saml20.xsl

Delegation module

The Delegation module is called DelegatorSTSModule.

Enables delegation to another module of the processing of an incoming token request. For example, IBM Tivoli Federated Identity Manager uses this module to enable security administrators to substitute a custom mapping module in place of the default mapping module.

The Delegation module must be configured to specify the module instances to receive the delegated action.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Other

Configuration properties

Module instance

The name of a module instance to which the action (validate, map, issue, exchange, or other) required by this location in the module chain should be delegated.

Module instance

The name of a module instance to which the action (validate, map, issue, exchange, or other) required by this location in the module chain should be delegated.

Digital Signature module

The Digital Signature module is called DSigSTSModule.

Enables the Secure Trust Service (STS) to validate signatures on incoming WS-Trust requests. Enables enhancement of WS-Trust transactions with message-level security.

The configuration properties specify the methods to use for validating the digital signature.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Validate

Configuration properties

Use Subject Key Identifier

Instructs the module to use the subject key identifier to locate an appropriate validation key.

The Subject Key Identifier uniquely identifies the entity to whom the certificate was issued. This value can be used by the trust service to determine which key from the IBM Tivoli Federated Identity Manager key service to use to validate the signature on the message.

Use Subject Distinguished Name (DN)

Instructs the module to use the subject distinguished name (DN) to select the validation key. This value instructs the IBM Tivoli Federated Identity Manager key service to use the distinguished name to determine which key to use to validate the signature.

Use Certificate included in Signature

Instructs the module to use the certificate included in the signature itself to validate the signature. This value instructs the trust service to examine the certificate that is included in the signature, and to validate that the certificate has been signed by a trusted certificate authority and has not been revoked. When the certificate is valid, the trust service calls the key service to validate the signature on the request.

Dynamic Chain Selection module

The Dynamic Chain Selection module is called DynamicChainSelectionModule.

Requests that the trust service invoke a dynamic chain after processing is finished on the current module chain.

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Other

Configuration properties

None.

Java Authentication and Authorization Service module

The Java Authentication and Authorization Service module is called JAASSTSModule.

Java Authentication and Authorization Service module. Used by IBM Tivoli Federated Identity Manager to obtain a valid JAAS subject from WebSphere Application Server for use in token exchange operations.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Validate

Configuration properties

Use domain-qualified principal names

Specifies whether to use a principal name that includes the user registry domain. For example, in an LDAP user registry the syntax is *LDAP_server:port_number\username*. For example:

```
localhost:389\elain
```

When this value is not specified, just the value of *username* is used.

Use of a domain-qualified principal name is typically necessary only when the deployment includes multiple user registries, and there is a need to differentiate between users from different domains. In most deployments, this value is not necessary.

Kerberos module

The Kerberos module is called KerberosSTSModule.

KerberosSTSModule validates Kerberos security tokens with a token type of:

```
http://schemas.xmlsoap.org/ws/2003/12/kerberos/
  Kerberosv5_AP_REQ
http://docs.oasis-open.org/wss/oasis-wss-kerberos-
  token-profile-1.1#Kerberosv5_AP_REQ
http://docs.oasis-open.org/wss/oasis-wss-kerberos-
  oken-profile-1.1#GSS_Kerberosv5_AP_REQ
```

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue

Configuration properties (Validate mode)

Kerberos service keytab file name

(Optional) Specify the path of the Kerberos service keytab file for the Kerberos service identified by the Kerberos security token. If this property is not specified, it must be specified as an STS universal user context attribute.

Configuration properties (Issue mode)

Kerberos realm name

Specify the name of the Kerberos realm. The Kerberos realm name must match the realm configured within krb5.conf. This field is optional. For example, EXAMPLE.COM. If this property is not specified, it must be specified as an STS universal user context attribute.

Kerberos service name

Specify the principal name of the Kerberos service in the form of service-name or service-name@realm. For example, krb5service/krb5host.example.com or krb5service@example.comkrb5service/krb5host.example

When the realm is not supplied, the default realm, if any, configured in krb5.conf is used.

The name must match the configuration in Active Directory or whatever Kerberos directory is being used.

Kerberos security token value types

Select the value type of the Kerberos binary security token to issue. The default value is

`http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ.`

Specifying the encryption level

The Kerberos STS module does not support DES encryption types.

Ensure that the setting for the service user does not require DES encryption.

1. On the Windows server, open the Account tab for the service user.
2. Clear the configuration option:
   `Use DES encryption types for this account`

The configuration option might be selected by default. If you leave the configuration option selected, an encryption error is thrown during the issuing of the Kerberos service ticket. The error states that the encryption algorithm is not supported.

Kerberos delegation module

The Kerberos delegation module is called KerberosDelegationSTSModule.

The Kerberos delegation module issues Kerberos credentials for a given user and service with a token type of:

`http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`

The module supports issue and exchange modes. The module facilitates the issuing of Kerberos Constrained Delegation application service tickets, also known as Service for User To Proxy (S4U2Proxy).

This module makes it possible for the Tivoli Access Manager WebSEAL to support Kerberos junctions. The junctions are WebSEAL junctions to a web server, such as IIS, that is configured for Integrated Windows Authentication (SPNEGO).

Deployment scenarios for this module type

- Custom trust chains
- Web services security management

Supported modes

- Issue
- Exchange

Configuration properties (initialization)

Maximum size of the user credential cache
This value determines the number of impersonation handles and user credentials cached for performance reasons in the dynamically loaded library loaded by the module.

Set this number to the approximate number of expected concurrent end users of the service for high-volume transactions. The higher the number, the more memory that might be consumed by the IBM Tivoli Federated Identity Manager runtime application.
Default: 100

Configuration properties (issue mode)

Default target Service Principal Name
(Optional) This entry is the default target Service Principal Name. This option is used for the WS-Trust clients that do not send the target Service Principal Name in the AppliesTo ServiceName element of the RST, and that

do not have a mapping rule to configure the target Service Principal Name as an STSUniversalUser context attribute.

Options for adding a Tivoli Access Manager username for Kerberos authentication

Use the options to specify whether the module auto-appends 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 are:

- Do not add a suffix to the username.

  This option retains the current user name.

- Add the machine DNS domain as a suffix to the username.

  This option automatically appends the DNS domain suffix for the IBM Tivoli Federated Identity Manager runtime machine to the principal name in the STSUniversalUser. Then, it prompts 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 for e-business 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: The module auto-appends the DNS domain name when the STSUniversalUser principal name does *not* already contain the @ character. This setting has no effect under the following circumstances:

  – If you use a mapping rule to append a suffix containing the @ character to the user principal name

  – If the Tivoli Access Manager for e-business user name contains an @ character

- Add the configured suffix to the username

  This option is used to optimize the module behavior for use in IBM Security Access Manager for Web configurations using Kerberos junctions.

  The administrator can use this option to manually specify the suffix. Use this option when the userPrincipalName attribute for the user does not match the DNS domain name of the Windows machine running the IBM 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`

Key Encryption and Signature Service STS module

The Security token service module for Key Encryption and Signature Service (KESS) is called the KESS STS Module.

This module facilitates the following generic XML security operations in an STS module:

- XML digital signing of a portion of XML
- XML signature validation of a signed portion of XML

- XML encryption of a portion of XML
- XML decryption of a portion of encrypted XML

The KESS STS module performs discretionary XML security operations as part of message workflows. With this module, users can automatically use the centralized key storage and hardware cryptography support.

Scenario

The following example describes a standard scenario:

- A user previously deployed IBM Tivoli Federated Identity Manager for federated single sign-on or identity mediation. The XML security keys are stored in KESS.
- The user wants to generate or validate (or both) additional XML security messages not natively supported by IBM Tivoli Federated Identity Manager. The user also wants to use the stored KESS security keys for the generation or validation (or both).
- The user writes specialized company-specific application code to generate the XML messages.
- The user uses the IBM Tivoli Federated Identity Manager STS and the stored keys to perform the XML security operation.

Deployment scenarios for this module type

Custom trust chains

Supported modes

Map

Configuration properties

Use the administration console to configure an XML security operation. The console prompts you to specify configuration properties. The following section lists the properties that you can configure.

Please select the operation to perform

Specifies the configured operation for the module: sign, validate, encrypt, or decrypt. Depending on the operation, select one operation.

STS Universal User Attribute

Specifies on which attribute in the `STSUU AttributeList` to perform the operation. This property is common to all operations.

Signature properties

Provides confirmation to the receiving party that a message was not altered during transmission. If performing the signature operation, select from the following configuration properties.

- **Select the key to use if performing signing**: Generates the XML signature with the keystore alias of the private key. Select the alias from the available keystores.
- **Include the Public Key?**: Specifies whether to include the public key in the `KeyInfo`.
- **Include X509 Subject Issuer Details?**: Specifies whether to include issuer information in the `KeyInfo/X509Data`.
- **Include the X509 Subject Name?**: Specifies whether to include the X509SubjectName (distinguished name of signing certificate) in the `KeyInfo/X509Data`.
- **Include the X509 Certificate Data?**: Specifies whether to include the `X509CertificateData` in the `KeyInfo/X509Data`.

- **Include the X509 Subject Key Identifier?**: Specifies whether to include the Subject Key Identifier in the `KeyInfo/X509Data`.
- **Use Inclusive Namespaces when signing**: Specifies whether to use the `InclusiveNamespaces` construct, which means employing exclusive XML canonicalization for greater standardization.

`Validation properties`

Confirms to the receiving party that the transmitted message came from a trusted source. If performing validation, select from the following configuration properties.

- **Validate signature on the STSUniversalUser attribute**: Specifies whether to enable signature validation.
- **Use the KeyInfo of the XML to find X.509 certificate for signature validation**: Checks if the signature came from an expected key. Selecting this flag includes specifying the Subject distinguished name expression for the allowable X.509 certificates. Enter a regular expression to validate the Subject distinguished name that is returned in the `KeyInfo`.
- **Use keystore alias to find public key for signature validation**: Specifies a key from the available keystores that can validate the signature.
- **Use Included X509 Cert Data (when validating signatures)**: Specifies whether to use embedded certificate data or a KESS keystore certificate to validate the signature. For example, the KESS keystore might receive a message signed by a key that is not in the KESS keystore. However, the signer of that key is in the keystore. Signature validation might then occur with the certificate embedded in the message.

`Encryption properties`

Provides an extra layer of security for messages that have been signed. Encryption uses the public key of the intended recipient. If performing encryption, select from the following configuration properties.

- **Select the key to use if performing encryption**: Generates XML encryption with the keystore alias of the public key. Select the alias from the available keystores.
- **Block Encryption Algorithm (encryption only)**: Specifies the block encryption algorithm. Select the identifier for one of the following block encryption algorithms:
  - Triple DES
  - AES 128
  - AES 192
  - AES 256
- **Key Transport Algorithm (encryption only)**: Specifies the key transport algorithm. Select the identifier for one of the following key transport algorithms:
  - RSA-1-5
  - RSA-OAEP-MGF1P
  - SHA 1

    Note: SHA 1 does not come preconfigured. To support this algorithm, you must configure other security providers.

`Decryption properties`

Specifies the private key for decrypting the encrypted message. If performing decryption, select the following configuration property.

Select the key to use if performing decryption
Generates XML decryption with the keystore alias of the private key. Select the alias from the available keystores.

Liberty 1.1 module

The Liberty 1.1 module is called Liberty11STSModule.

Note: Liberty protocol is being deprecated in the Tivoli Federated Identity Manager 6.2.2 release.

Used for Liberty profiles for Liberty v1.1 federations. Liberty token modules call out to the IBM Tivoli Federated Identity Manager alias service for alias (name identifier) lookups.

Deployment scenarios for this module type

- Single sign-on federations

  Liberty 1.1 only

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Issue or Exchange mode

Number of seconds before the issue date that an assertion is considered valid.
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Amount of time the assertion is valid after being issued (seconds)
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Include the following attribute types (a "*" means include all types)
Specifies the types of attributes to be inserted during token creation. The attributes consist of information about the identity (user). Use the && character to separate attribute types. By default all types are supported, as indicated by the asterisk (*) wildcard.

For example, to add user-defined attribute types *type1* and *type2*, enter:

```
type1&&type2
```

Liberty 1.2 module

The Liberty 1.2 module is called Liberty12STSModule.

Note: Liberty protocol is being deprecated in the Tivoli Federated Identity Manager 6.2.2 release.
Used for Liberty profiles for Liberty v1.2 federations. Liberty token modules call out to the IBM Tivoli Federated Identity Manager alias service for alias (name identifier) lookups.

Deployment scenarios for this module type

- Single sign-on federations
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Validate mode

Username to be used for anonymous users

A user can access a service anonymously through this one-time name identifier. The user name entered here is one that the service provider recognizes as a one-time name identifier for a legitimate user in the local user registry.

Users can access a resource on the service provider without establishing a federated identity through this feature. This is useful when the service provider does not need to know the identity of the user account, but must know if the identity provider has authenticated, and can vouch for the user.

Note: The user identity must exist as a valid user in the user registry. This property is set only when adding an identity provider partner.

Issue mode

Number of seconds before the issue date that an assertion is considered valid.

Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Amount of time the assertion is valid after being issued (seconds)

Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Include the following attribute types (a "*" means include all types)

Specifies the types of attributes to be inserted during token creation. The attributes consist of information about the identity (user). Use the && character to separate attribute types. By default all types are supported, as indicated by the asterisk (*) wildcard.

For example, to add user-defined attribute types *type1* and *type2*, enter:

```
type1&&type2
```

LTPA module

The LTPA module is called STSLTPATokenModule.

Validates and issues LTPAv1 and LTPAv2 tokens. These are represented as BinarySecurityToken elements.

This module does not support the initial generation of LTPA keys. You must provide a set of LTPA keys that have been generated by another source such as a WebSphere application server.

You can use the WebSphere administration console to export LTPA keys. On WebSphere 6.1, access the following menu:

Security > Secure administrations, applications, and infrastructure > Authentication mechanisms and expiration. Enter a password and a filename. Click **Export keys**. You must remember the password you specify because you must re-enter it on the IBM Tivoli Federated Identity Manager console. The password is used to protect the private key in the key file.

Deployment scenarios for this module type

- Federated single sign-on
- Web services security management
- Custom trust chain

Supported modes

- Validate
- Exchange
- Issue

Configuration properties

Validate mode

Location of LTPA File
(Required) The fully-qualified path name for the LTPA file you want to use. Optionally, use the **Browse** button to locate the file.

Import LTPA File
Use this button to import the file that you specified in the **Location of LTPA File** field. You must click this button to use the file. After you import the file, the contents shows in the **File contents of the imported LTPA file** field.

File contents of the imported LTPA file
Shows the contents of the LTPA file you imported. You cannot update the file contents in this field. It is for informational purposes only.

Password for key protection
(Required) The password that was used to protect the keys created by the partner. It must be the same password that was used when the keys were created by the partner.

Password for key protection (confirm)
Enter the password for key protections again for verification.

Use the FIPS standard
Check to enable the Federal Information Processing Standards (FIPS). If FIPS was enabled when you created your partner, check this box. The default is unchecked.

Issue or Exchange mode

Location of LTPA File
(Required) The fully-qualified path name for the LTPA file you want to use. Optionally, use the **Browse** button to locate the file.

Import LTPA File
Use this button to import the file that you specified in the **Location of LTPA File** field. You must click this button to use the file. After you import the file, the contents shows in the **File contents of the imported LTPA file** field.

File contents of the imported LTPA file
Shows the contents of the LTPA file you imported. You cannot update the file contents in this field. It is for informational purposes only.

Password for key protection
(Required) The password that was used to protect the keys created by the partner. It must be the same password that was used when the keys were created by the partner.

Password for key protection (confirm)
Enter the password for key protection again for verification.

Use the FIPS standard
Check to enable the Federal Information Processing Standards (FIPS). If FIPS was enabled when you created your partner, check this box. The default is unchecked.

Number of minutes before the created token expires
(Required) Indicates how long, from the time of token creation, the token remains valid. Specify the value in minutes. You can override this value using the expiration Principle value in the Universal User. The default value is 120 minutes.

Realm used to create the user ID
The realm name to append to the user ID during token creation. You can override this value using the realm Principle value in the Universal User. If you do not specify a name here, then it is assumed that the PrincipleName in the Universal User document is already in the following form: *realm:port/userID*

Version of LTPA token to issue
The version number of the LTPA token you are issuing. Select 1 or 2 from the drop-down list, denoting LTPA Version 1 or Version 2.

Attributes to add to a version 2 token
Specify the types of attributes to include in the assertion. Use this field only for LTPA Version 2 tokens. An 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 in the assertion, type `urn:oasis:names:tc:SAML:2.0:assertion` in the box. Use && to separate multiple attribute types.

PassTicket module

The PassTicket module is called PassTicketSTSModule.

Issues and validates Resource Access Control Facility (RACF®) PassTicket tokens. PassTicket tokens extend the structure of Username tokens by adding a generated PassTicket. You can use PassTicket tokens in non-z/OS environments.

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Validate mode

The application name used for PassTicket generation and validation

The name of the application that was used to generate the unique PassTicket. This must be an 8 character user ID. The characters must be alphanumeric. For example:

`G1SGRAM`

Amount of time the token is valid after being issued (seconds, enter -1 for no expiration)

An integer value indicating the amount of time, in seconds, that the token remains valid. There are no minimum or maximum values imposed.

Default value: `300`

The special value -1 means that the token does not expire.

Enable signature validation

Enables or disables validation of signatures in the token module. Select the check box to enable signature validation.

Select validation key

Specify the validation key that the partner must use.

Keystore

The keystore containing the key or certificate to be used. The menu shows the keystores that have been configured into the key service. Select the appropriate keystore.

Keystore Password

The password for the keystore.

List Keys

Shows a table that lists the keys contained within the specified keystore. Select the radio button in the Select column to specify the key or certificate to use.

Issue mode

Include nonce in token

Includes a nonce (random bits used for the purpose of obfuscating the element) in the token.

Include token creation time in token

Adds a timestamp to the token, indicating the creation time of the token.

The application name used for PassTicket generation and validation

The name of the application that was used to generate the unique PassTicket. This must be an 8 character user ID. The characters must be alphanumeric. For example:

G1SGRAM

PassTicket key
A key value consisting of exactly 16 hexadecimal digits, to be used to generate a valid PassTicket.

Enable the signing of RACF PassTicket tokens
Enables or disables the signing of the PassTicket token module. Select the check box to enable signature validation.

Select key for signing tokens
Specify the key to use to sign the outgoing tokens.

Keystore
The keystore containing the key or certificate to be used. The menu shows the keystores that have been configured into the key service. Select the appropriate keystore.

Keystore Password
The password for the keystore.

List Keys
Shows a table that lists the keys contained within the specified keystore. Select the radio button in the Select column to specify the key or certificate to use.

SAML 1.0 module

The SAML 1.0 module is called SAML10STSModule.

Used for single sign-on in SAML 1.0 federations.

Deployment scenarios for this module type

- Single sign-on federations
- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties
Validate mode

Enable one-time assertion use enforcement
Specifies whether this artifact (token) is used only once.

Enable Signature Validation
Enables or disables validation of signatures in the token module. Select the check box to enable signature validation.

Select validation key
Specifies the validation key that the partner must use.

Use XML signature's KeyInfo to find X.509 certificate for signature validation
Determines the appropriate certificate for signature validation. When you select this option, you must provide the subject distinguished name that matches the certificate.

Use keystore alias to find public key for signature validation
Specifies a public key for signature validation, which is the default. Select the keystore, password, and appropriate key from the list.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured for the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Specify the Subject DN expression for the allowable X.509 certificates
Specifies the expression for the X.509 certificate used for validation. The key used to sign the message must have a distinguished name that matches this value. The key must also match a certificate in one of the IBM Tivoli Federated Identity Manager keystores.

The message signature must include one or more of the following elements that point to this certificate:

- X509 SubjectkeyIdentifier element
- X509 Certificate
- X509 SubjectName
- Public key

Following are examples of a distinguished name:

- `cn=fimdemo.ibm.com,ou=tameb,o=tivoli,c=us`
- `REGEXP:cn=fimdemo.*`

  The key used to sign the message must have a distinguished name that starts with `cn=fimdemo`.

Create multiple attribute statements in the Universal User
Specifies whether to keep multiple attribute statements in the groups in which they were received. This option might be necessary if your custom identity mapping rules operate on one or more specific groups of attribute statements.

If you do not select this check box, multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. The default setting of the check box is not selected. This setting is appropriate for most configurations.

Issue mode

The name of the organization issuing the assertions
Shows a string specifying the name of the organization (for example, a company) that issues the SAML assertions.

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.

This field must contain a value.

Amount of time the assertion is valid after being issued (seconds)
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Sign SAML Assertions
Specifies whether SAML assertions must be signed.

Select Key for Signing Assertions
Specifies the key to use when signing SAML assertions:

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Select the KeyInfo elements to include
Determines what `KeyInfo` elements to include in the digital signature when signing a SAML message or assertion. Select one or more of the following elements.

Include the X509 certificate data?
Specifies whether to include the BASE64 encoded certificate data with your signature. Select **Yes** (default) to include the X.509 certificate data, and **No** to exclude the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Name?
Specifies whether to include the subject name with your signature. Select **No** (default) to exclude the X.509 subject name, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Key Identifier?
Specifies whether to include the X.509 subject key identifier with your signature. Select **No** (default) to exclude the subject key identifier, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include X509 Subject Issuer Details?
Specifies whether to include the issuer name and the certificate serial number with your signature. Select **No** (default) to exclude the X.509 subject issuer details, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the Public Key?
Specifies whether to include the public key with your signature. Select **No** (default) to exclude the public key, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Note: If you specify **No** for all five `KeyInfo` elements, `X509Certificate` data is still included in the signature by default.

Signature Algorithm for signing SAML Assertions
Specifies the signature algorithm to use to sign the SAML assertion.

RSA-SHA1
http://www.w3.org/2000/09/xmldsig#rsa-sha1

DSA-SHA1
http://www.w3.org/2000/09/xmldsig#dsa-sha1

RSA-SHA256
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

Note: The chosen signature algorithm must match the signing key type that was set in the federation level to prevent a signature failure. For example, select DSA-SHA1 for DSA keys.

Include the following attribute types
Specifies the types of attributes to include in the assertion. The default asterisk (*) setting includes all the attribute types that are specified in the identity mapping file or by the custom mapping module.

To specify one or more attribute types individually, enter each attribute type in the field. Use && to separate multiple attribute types. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:1.0:assertion` in the assertion, enter `urn:oasis:names:tc:SAML:1.0:assertion` in the field.

Subject Confirmation Method
Specifies the subject confirmation method for the assertion. If you select the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. `STSUniversalUser` can provide the data for the `SubjectConfirmationMethod KeyInfo`. The data can also be extracted from the signed request data.

Valid values can be:

- No Subject Confirmation Method (blank)
- urn:oasis:names:tc:SAML:1.0:bearer
- urn:oasis:names:tc:SAML:1.0:holder-of-key
- urn:oasis:names:tc:SAML:1.0:sender-vouches

For the SubjectConfirmationMethod (SCM) to be issued correctly, the client must sign the RST request and include a KeyInfo that can be used for the SCM when sending the RequestSecurityToken (RST). To use the holder-of-key capability, the XSLT mapping rules must be updated.
For example:

```
<stsuuser:AttributeList>
 <stsuuser:Attribute name="SamlSubjectConfirmationMethod"
 type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
 </stsuuser:Value>
 </stsuuser:Attribute>
</stsuuser:AttributeList>
```

SAML 1.1 module

The SAML 1.1 module is called SAML11STSModule.

Used for WS-Federation single sign-on federations.

Deployment scenarios for this module type

- Single sign-on federations
- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Validate mode

Enable one-time assertion use enforcement

Specifies whether to use this artifact or token only once.

Enable Signature Validation

Enables or disables validation of signatures in the token module. Select the check box to enable signature validation.

Select validation key

Specifies the validation key that the partner must use.

Use XML signature's KeyInfo to find X.509 certificate for signature validation

Determines the appropriate certificate for signature validation. When you select this option, you must provide the subject distinguished name that matches the certificate.

Use keystore alias to find public key for signature validation

Specifies a public key for signature validation, which is the default. Select the keystore, password, and appropriate key from the list.

Keystore

Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured for the key service.

Keystore Password

Specifies the password for the keystore.

List Keys

Shows a table that lists the keys in the specified keystore. Select the radio button in the **Select** column to specify the key or certificate.

Specify the Subject DN expression for the allowable X.509 certificates

Specifies the expression for the X.509 certificate used for validation. The key used to sign the message must have a distinguished name that matches this value. The key must also match a certificate in one of the IBM Tivoli Federated Identity Manager keystores.

The message signature must include one or more of the following elements that point to this certificate:

- X509 SubjectkeyIdentifier element
- X509 Certificate
- X509 SubjectName
- Public key

Examples of a distinguished name:

- `cn=fimdemo.ibm.com,ou=tameb,o=tivoli,c=us`
- `REGEXP:cn=fimdemo.*`

  The key used to sign the message must have a distinguished name that starts with `cn=fimdemo`.

Create multiple attribute statements in the Universal User

Specifies whether to keep multiple attribute statements in the groups in

which they were received. 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 you do not select this check box, multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. The default setting of the check box is not selected. This setting is appropriate for most configurations.

Issue mode

The name of the organization issuing the assertions
Shows a string specifying the name of the organization (for example, a company) that issues the SAML assertions.

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.

This field must contain a value.

Amount of time the assertion is valid after being issued (seconds)
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Sign SAML Assertions
Specifies whether SAML assertions must be signed.

Select Key for Signing Assertions
Specifies the key to use when signing SAML assertions:

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Select the KeyInfo elements to include
Determines what `KeyInfo` elements to include in the digital signature when signing a SAML message or assertion. Select one or more of the following elements.

Include the X509 certificate data?
Specifies whether to include the BASE64 encoded certificate data with your signature. Select **Yes** (default) includes the X.509 certificate data, and **No** to exclude the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Name?
Specifies whether to include the subject name with your signature. Select **No** (default) to exclude the X.509 subject name, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Key Identifier?
Specifies whether to include the X.509 subject key identifier with your signature. Select **No** (default) to exclude the subject key identifier, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include X509 Subject Issuer Details?
Specifies whether to include the issuer name and the certificate serial number with your signature. Select **No** (default) excludes the X.509 subject issuer details, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the Public Key?
Specifies whether to include the public key with your signature. Select **No** (default) to exclude the public key, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Note: If you specify **No** for all five `KeyInfo` elements, `X509Certificate` data is still included in the signature by default.

Signature Algorithm for signing SAML Assertions
Specifies the signature algorithm to use to sign the SAML assertion.

RSA-SHA1
`http://www.w3.org/2000/09/xmldsig#rsa-sha1`

DSA-SHA1
`http://www.w3.org/2000/09/xmldsig#dsa-sha1`

RSA-SHA256
`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`

Note: The chosen signature algorithm must match the signing key type that was set in the federation level to prevent a signature failure. For example, select DSA-SHA1 for DSA keys.

Include the InclusiveNamespaces element in the canonicalization of the assertion during signature creation
Specifies whether to use the `InclusiveNamespaces` construct, which means employing exclusive XML canonicalization for greater standardization. The default is cleared.

Include the following attribute types
Specifies the types of attributes to include in the assertion. The default asterisk (*) setting includes all the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, enter each attribute type in the field. Use && to separate multiple attribute types. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:1.0:assertion` in the assertion, enter `urn:oasis:names:tc:SAML:1.0:assertion` in the field.

Subject Confirmation Method
Specifies the subject confirmation method for the assertion. If you select the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. `STSUniversalUser` can provide the data for the `SubjectConfirmationMethod KeyInfo`. The data can also be extracted from the signed request data.

For the SubjectConfirmationMethod (SCM) to be issued correctly, the client must sign the RST request and include a KeyInfo that can be used for the

SCM when sending the RequestSecurityToken (RST). To use the holder-of-key capability, the XSLT mapping rules must be updated. For example:

```
<stsuuser:AttributeList>
 <stsuuser:Attribute name="SamlSubjectConfirmationMethod"
 type="urn:oasis:names:tc:SAML:1.0:assertion">
  <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
 </stsuuser:Value>
 </stsuuser:Attribute>
</stsuuser:AttributeList>
```

SAML 2.0 module

The SAML 2.0 module is called Saml20STSTokenModule.

Used for single sign-on in SAML 2.0 federations.

Deployment scenarios for this module type

- Single sign-on federations
- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Validate mode

Enable one-time assertion use enforcement
Specifies whether to use the artifact or token only once.

Enable Signature Validation
Enables or disables validation of signatures in the token module. Select the check box to enable signature validation.

Select validation key
Specifies the validation key that the partner must use.

Use XML signature's KeyInfo to find X.509 certificate for signature validation
Determines the appropriate certificate for signature validation. When you select this option, you must provide the subject distinguished name that matches the certificate.

Use keystore alias to find public key for signature validation
Specifies a public key for signature validation, which is the default. Select the keystore, password, and appropriate key from the list.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured for the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Specify the Subject DN expression for the allowable X.509 certificates

Specifies the expression for the X.509 certificate used for validation. The key used to sign the message must have a distinguished name that matches this value. The key must also match a certificate in one of the IBM Tivoli Federated Identity Manager keystores.

The message signature must include one or more of the following elements that point to this certificate:

- X509 SubjectkeyIdentifier element
- X509 Certificate
- X509 SubjectName
- Public key

Examples of a distinguished name:

- `cn=fimdemo.ibm.com,ou=tameb,o=tivoli,c=us`
- `REGEXP:cn=fimdemo.*`

The key used to sign the message must have a distinguished name that starts with `cn=fimdemo`.

Select a decryption key

Select the key to use to decrypt encrypted messages.

Keystore

Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password

Specifies the password for the keystore.

List Keys

Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Create multiple attribute statements in the Universal User

Specifies whether to keep multiple attribute statements in the groups in which they were received. 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 you do not select this check box, multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. The default setting of the check box is not selected. This setting is appropriate for most configurations.

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.

Default NameID Format for Assertion validation

Specifies a parameter for use during validation of a SAML assertion. The parameter is used to determine processing rules for the NameID element when one of the following conditions exists:

- If there is not an explicit Format attribute included in the assertion
- If the Format attribute is urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

Typically this parameter is needed only for STS chains that process SAML assertions that don't set the Format attribute. A normal example value is: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress.

Issue mode

The name of the organization issuing the assertions
Shows a string specifying the name of the organization (for example, a company) that issues the SAML assertions.

Number of seconds before the issue date that an assertion is considered valid (seconds).
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Amount of time the assertion is valid after being issued (seconds)
Default: 60 seconds

There is no minimum or maximum value enforced.

This field must contain a value.

Include the following attribute types (a "*" means include all types)
Specifies the types of attributes to be inserted during token creation. The attributes consist of information about the identity (user). Use the && character to separate attribute types. By default, all types are supported, as indicated by the asterisk (*) wildcard.

For example, to add user-defined attribute types *type1* and *type2*, enter:

```
type1&&type2
```

Sign SAML Assertions
Specifies whether SAML assertions must be signed.

Select Key for Signing Assertions
Specifies the key to use when signing SAML assertions.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Select the KeyInfo elements to include
Determines what `KeyInfo` elements to include in the digital signature when signing a SAML message or assertion. Select one or more of the following elements.

Include the X509 certificate data?
Specifies whether to include the BASE64 encoded certificate data with your signature. Select **Yes** (default) to include the X.509 certificate data, and **No** to exclude the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Name?
Specifies whether to include the subject name with your signature.

Select **No** (default) to exclude the X.509 subject name, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the X509 Subject Key Identifier?
Specifies whether to include the X.509 subject key identifier with your signature. Select **No** (default) to exclude the subject key identifier, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include X509 Subject Issuer Details?
Specifies whether to include the issuer name and the certificate serial number with your signature. Select **No** (default) to exclude the X.509 subject issuer details, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Include the Public Key?
Specifies whether to include the public key with your signature. Select **No** (default) to exclude the public key, and **Yes** to include the data. To change the default for this element, change it in the custom properties.

Note: If you specify **No** for all five `KeyInfo` elements, `X509Certificate` data is still included in the signature by default.

Signature Algorithm for signing SAML Assertions
Specifies the signature algorithm to use to sign the SAML assertion.

RSA-SHA1
`http://www.w3.org/2000/09/xmldsig#rsa-sha1`

DSA-SHA1
`http://www.w3.org/2000/09/xmldsig#dsa-sha1`

RSA-SHA256
`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`

Note: The chosen signature algorithm must match the signing key type that was set in the federation level to prevent a signature failure. For example, select DSA-SHA1 for DSA keys.

Select the key for encrypting assertion elements for this partner
Specifies the key to use to encrypt assertions.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Encrypt assertions
Specifies whether assertions are to be encrypted. If selected, you must specify an encryption key.

Encrypt assertion Attribute elements.
Specifies whether Attribute elements within the assertions are to be encrypted. If selected, you must specify an encryption key.

Encrypt NameID elements in assertions.
Specifies whether `NameID` elements in the assertions are to be encrypted. If selected, you must specify an encryption key.

Block Encryption algorithm
Specifies the encryption algorithm to use to encrypt data for this partner.

Triple DES
Triple Digital Encryption Standard

AES-128
Advanced Encryption Standard 128-bit

AES-192
Advanced Encryption Standard 192-bit

AES-256
Advanced Encryption Standard 256-bit

Subject Confirmation Method
Specifies the subject confirmation method for the assertion. You can select one or more subject confirmation methods at the same time, or choose not to select any confirmation method. If you select the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. `STSUniversalUser` can provide the data for the `SubjectConfirmationMethod KeyInfo`. The data can also be extracted from the signed request data.

Valid values can be:

- No subject confirmation method (blank)
- urn:oasis:names:tc:SAML:1.0:bearer
- urn:oasis:names:tc:SAML:1.0:holder-of-key
- urn:oasis:names:tc:SAML:1.0:sender-vouches

You can use the identity mapping rules to add SubjectConfirmation information to the `STSUniversalUser`. See the example of an XSLT mapping rule with multiple subject confirmation methods.

```
<stsuuser:Attribute name="SamlSubjectConfirmationMethod"
 type="urn:oasis:names:tc:SAML:2.0:assertion">
  <stsuuser:Value>urn:oasis:names:tc:SAML:2.0:cm:bearer
   </stsuuser:Value>
  <stsuuser:Value>urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
   </stsuuser:Value>
 </stsuuser:Attribute>
```

Another way to add SubjectConfirmation information is through the response files. See the topic on “SAML 2.0 token module response file” on page 82.

Note: The values set in the identity mapping rule takes precedence over the settings in the response file. If there are different values for the mapping rule and response file, the assertion contains the values which was set in the mapping rule.
For the SubjectConfirmationMethod to be issued correctly, the client must sign the RST request and include a KeyInfo used for the SCM when

sending the RequestSecurityToken (RST). To use the holder-of-key capability, the XSLT mapping rules must be updated.
For example:

```
<stsuuser:AttributeList>
 <stsuuser:Attribute name="SamlSubjectConfirmationMethod"
   type="urn:oasis:names:tc:SAML:2.0:assertion">
  <stsuuser:Value>urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
  </stsuuser:Value>
 </stsuuser:Attribute>
</stsuuser:AttributeList>
```

Security token service message logger module

The Security token service message logger module is called STSMessageLoggerModule.

The STSMessageLoggerModule tracks all or part of a Security Token Service transaction, including requests, responses, incoming and outgoing mappings, errors and exceptions.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Other

Configuration properties (Federation)

Enable STS Message logging to log file
Specify whether to enable this instance of the logging module in the current STS chain to a log file. If checked, you can also select the type of logging to enable, such as requests, responses, and identity mapping logging. Then, specify the log file in the **Log name** field. You can also specify the maximum log file size and maximum number of log files if you choose this type of logging.

The default is checked.

Enable STS Message logging to custom extension
Specify whether to enable this instance of the logging module in the current STS chain to a custom extension. If checked, you can also select the type of logging to enable, such as requests, responses, and identity mapping logging. Then, specify the custom logger in the **Custom Logger Extension ID** field.

The default is unchecked.

Log name
The name of the log file where the data is saved. A unique log file name for different instances or for the federation and partner creates a logger to handle the logging of that particular configuration.

The default value is *STSmessages*.

This field is applicable for log files only.

Only alphanumeric values are allowed. White space is not allowed.

Enable log on error only
Specifies whether to enable logging of only the errors.

The default is disabled.

Enable log requests
Specifies whether to enable logging of request information coming in to this module chain.

The default is enabled.

Enable log responses
Specifies whether to enable logging of response information coming in to this module chain.

The default is enabled.

Enable log identity mappings
Specifies whether to enable logging of the identity mappings (STSUniversalUser), which include the incoming identity and outgoing mapped, or transformed, identity. It logs only the STSUniversalUser information when invoking a module that has the mode of 'map'.

The default is disabled.

Maximum log file size (megabytes)
The maximum size, in megabytes, of the log file. When this size is reached, the file rolls over. Rolling over involves saving the log file with a different name, and continues logging with a new, empty log file with the original name.

This field is applicable for log files only.

The default value is 10.

Specifying 0 sets the maximum size to unlimited.

The allowable range of values for this field is from 0 to 209715.

Maximum number of log files
The maximum number of log files to retain at a given time.

This field is applicable for log files only.

The default value is 10.

The allowable range of values for this field is from 1 to 32767.

Custom Logger Extension ID
The name of the custom logger extension ID that you defined.

This field is applicable for custom loggers only.

Configuration properties (Partner)

STS Message logging
Specify whether to enable this instance of the logging module in the current STS chain. If enabled, requests, responses, and identity mapping logging takes place. From the drop-down box, you can choose to use the same value as set by the Federation properties or change it, by selecting **enable** or **disable**.

The default is **Use federation property value**.

Log name
The name of the log file where the data is saved. A unique log file name for different instances, or for the federation and partner creates a logger to handle the logging of that particular configuration.

If no log name is specified, the federation log name is used.

Only alphanumeric values are allowed. White space is not allowed.

Log on error only
Specifies whether to enable logging of only the errors. From the drop-down box, you can choose to use the same value as set by the Federation properties or change it, by selecting **enable** or **disable**.

The default is **Use federation property value**.

Log requests
Specifies whether to enable logging of request information coming in to this module chain. From the drop-down box, you can choose to use the same value as set by the Federation properties or change it, by selecting **enable** or **disable**.

The default is **Use federation property value**.

Log responses
Specifies whether to enable logging of response information coming in to this module chain. From the drop-down box, you can choose to use the same value as set by the Federation properties or change it, by selecting **enable** or **disable**.

The default is **Use federation property value**.

Log identity mappings
Specifies whether to enable logging of the identity mappings (STSUniversalUser), which include the incoming identity and outgoing mapped, or transformed, identity. It logs only the STSUniversalUser information when invoking a module that has the mode of 'map'. From the drop-down box, you can choose to use the same value as set by the Federation properties or change it, by selecting **enable** or **disable**.

The default is **Use federation property value**.

Maximum log file size (megabytes)
The maximum size, in megabytes, of the log file. When this size is reached, one of the following takes place:

- The log file rolls over, where the number of log files is set to zero.
- The log file number is incremented, up to the maximum number of log files.

If no maximum log file size is specified, the federation maximum log file size is used.

Specifying 0 sets the maximum size to unlimited.

The allowable range of values for this field is from 0 to 209715.

Maximum number of log files
The maximum number of log files to retain at a given time.

If no maximum number of log files is specified, the federation maximum number of log files is used.

The allowable range of values for this field is from 1 to 32767.

Security token service universal user module

The Security token service universal user module is called STSUUSTSModule.

This module acts as a pass-through module to either pass in or out an XML-based STSUniversalUser token.

This module is useful for testing other STS modules or for simple custom trust client applications. It provides a simple means to directly call the trust service to issue more complex token types without having to first pass in another token, and then perform a mapping operation.

For example, when a SAML 1.0 assertion is needed (to send to an authenticated Web application), and the current username and attribute information about the user is known, use of the STSUniversal User module eliminates the need to create a token, validate it, perform a mapping operation, and then issue the SAML 1.0 assertion. With the STSUniversalUser STS Module, you can create a simple chain with only two elements:

- Validate STSUniversalUserToken
- Issue signed SAML 1.0 Assertion

The input STSUniversalUser token can contain the username, any extended attributes, and any attributes required for issuing the SAML assertion, as generated by the caller of the trust service.

Note: No mapping step is required.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Validate
- Exchange
- Issue

Configuration properties

None.

Tivoli Access Manager authentication module

The Tivoli Access Manager for e-business authentication module is called TAMAuthenticationSTSModule.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

This module uses the supplied credentials to do an Tivoli Access Manager for e-business authentication check. The module is used by the Web services security management component when performing authentication decisions on Web service requests. It can also be used by custom trust chains.

Note: This module is not a generic authentication module. It processes only authentication requests against Tivoli Access Manager for e-business.

Deployment scenarios for this module type

- Custom trust chains
- Web services security management

Supported modes

- Authenticate

Configuration properties

None.

Tivoli Access Manager for e-business authorization module

The Tivoli Access Manager for e-business authorization module is called TAMAuthorizationSTSModule.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

This module uses the supplied user identity and resource name to do an Tivoli Access Manager for e-business authorization check. The Web services security management component uses the module when authorizing decisions on Web service requests. It can also be used by custom trust chains.

Note: This module is not a generic authorization module. It processes only authorization requests against Tivoli Access Manager for e-business.

Deployment scenarios for this module type

- Custom trust chains
- Web services security management

Supported modes

- Authorize

Configuration properties

Stop the chain if unauthorized

Specifies to stop the trust service module chain execution if the user is not authorized. If the user is unauthorized and the check box is not selected, then the trust service chain execution continues. If the user is authorized, then this parameter is ignored.

Tivoli Access Manager for e-business credential module

The Tivoli Access Manager for e-business credential module is called IVCredModule.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

The trust service can create and use local tokens in an environment that is protected by Tivoli Access Manager for e-business. The support for Tivoli Access Manager for e-business credentials means that the trust service can also use the credentials for authorization decisions.

Deployment scenarios for this module type

- Single sign-on federations
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties

Validate mode

Enable signature validation
Enables or disables validation of signatures in the token module. Select the check box to enable signature validation.

Select validation key
Specifies the validation key that the partner must use.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Issue mode

Include the following attribute types (a "*" means include all types)
Specifies the types of attributes to be inserted during token creation. The attributes consist of information about the identity (user). Use the && character to separate attribute types. By default all types are supported, as indicated by the asterisk (*) wildcard.

For example, to add user-defined attribute types *type1* and *type2*, enter:

```
type1&&type2
```

Enable signatures
Specifies that signatures must be added to tokens.

Select signing key
Specifies the key to use to sign tokens.

Keystore
Specifies the keystore containing the key or certificate. The menu shows the keystores previously configured into the key service.

Keystore Password
Specifies the password for the keystore.

List Keys
Shows a table that lists the keys in the specified keystore. Select the radio button in the Select column to specify the key or certificate.

Select the KeyInfo elements to include
Specifies the elements of the signing certificate in the extended attributes of the credential. These attributes are only included if signatures are enabled. The default is for them to be disabled.

Include the Public Key?
Specifies whether to include this attribute. **No**, the default, excludes the public key. If **Yes**, the public key of the signing certificate is included in the Base64 encoded form. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_PUBKEY`.

Include the X509 Subject Name?
Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Name. If **Yes**, the distinguished name of the subject for the signing certificate is included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_SUBJECT`.

Include X509 Subject Issuer Details?
Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Issuer Details. If **Yes**, the issuer details of the signing certificate are included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_ISSUER`.

Include the X509 Subject Key Identifier?
Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Key Identifier. If **Yes**, the subject key identifier of the signing certificate is included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_SKI`.

Include the X509 Certificate Data?
Specifies whether to include this attribute. **No**, the default, excludes the X509 Certificate Data. If **Yes**, the certificate data of the signing certificate is included in the Base64 encoded form. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE`.

Note: If you specify **No** for all five `KeyInfo` elements, `X509Certificate` data is still included in the signature by default.

Tivoli Access Manager for e-business Global Signon Lockbox module

The Tivoli Access Manager for e-business Global Signon Lockbox module is called STSTAMGSOModule.


The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

This module retrieves a user name and password pair from the Tivoli Access Manager for e-business user registry for Tivoli Access Manager for e-business global signon (GSO) resources and sets them as attributes into the Principal section of the **`STSUniversalUser`**.

An example of an STSUU after this module has successfully retrieved a user name and password from the Tivoli Access Manager for e-business GSO Lockbox is:

```
<stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
  <stsuuser:Principal>
    <stsuuser:Attribute name="Password"
    type="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-username-token-profile-1.0#PasswordText">
      <stsuuser:Value>targetpwd</stsuuser:Value>
    </stsuuser:Attribute>
    <stsuuser:Attribute name="name"
     type="http://docs.oasis-open.org/wss/2004/01/
           oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <stsuuser:Value>targetuser</stsuuser:Value>
    </stsuuser:Attribute>
    <stsuuser:Attribute name="Username"
     type="http://docs.oasis-open.org/wss/2004/01/
```

```
          oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <stsuuser:Value>targetuser</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:Principal>
  ...
</stsuuser:STSUniversalUser>
```

Note: In the preceding example **STSUniversalUser**, the `type=` lines have been split for readability. The `type` value must be one continuous line.

A user can use the Tivoli Access Manager for e-business global signon technology to have an indexed set of additional user name and password pairs. Each user name and password pair is indexed through a string called a *target resource name*. The resource name relates to the target system or database which requires a separate login for access. When an Tivoli Access Manager for e-business wants to access that resource, the GSO lockbox is consulted to retrieve the required user name and password.

The target resource name is specified during the configuration of the trust module for Tivoli Access Manager for e-business Global Signon Lockbox.

Deployment scenarios for this module type

- Custom trust chains

Supported modes

- Map

Configuration properties

Tivoli Access Manager GSO Resource Name
The name of the Tivoli Access Manager for e-business resource that is GSO-enabled. This parameter is the value that you defined when you created the resource name in IBM Security Access Manager for Web. This field is required. There is no default value.

Overriding the configuration property

It is possible to override the configured Tivoli Access Manager for e-business GSO Resource Name at runtime. You can override the configuration by specifying setting an attribute in the STSUniversalUser. In the ContextAttributes container, specify a value for `targetResource`.

When this attribute is not present in the STSUniversalUser, the value stored for the configuration property Tivoli Access Manager for e-business GSO Resource Name is used.

Pool size
This property establishes the number of connections (PDContexts) to IBM Security Access Manager for Web that each configuration manages. Each configured chain creates its own set of connections.

The default is 2.

This value is not set through a graphical user interface. The default can be overridden by setting the custom runtime parameter tamgso.PDContextPoolSize.

Only one thread can use a PDContext at a time. This means that the pool size determines the maximum number of threads which can concurrently run this module in the chain. To increase the maximum number of threads, use the custom runtime parameter to increase the pool size.

Note: The value of this custom runtime parameter applies to *all* chains in the runtime.

Tivoli Access Manager configuration file
This property points to an Tivoli Access Manager for e-business configuration file, for initializing the Tivoli Access Manager for e-business administration API. This property is set in the administration console when an instance of the security token service module is created. The default value is **amconfig.conf**.

The configuration file is created when the IBM Tivoli Federated Identity Manager runtime is configured. In most cases, you do not have to modify this value.

Tivoli Directory Integrator module

The Tivoli Directory Integrator module is called DirectoryIntegratorSTSModule.

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.

Deployment of this module involves configuration of Tivoli Directory Integrator. For more information, see the *IBM Tivoli Federated Identity Manager Installation Guide*.

Deployment scenarios for this module type

- Custom trust chains
- Single sign-on

Supported modes

- Map

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`.

Select the identification format for the Work Entry attributes
Specify the Work Entry attribute identification format.

Attribute name
The TDI Work Entry will use the name to identify its attributes.

Attribute name and attribute type
The TDI Work Entry will use both the name and type to identify its attributes. Use this method if multiple attributes with the same name exist.

Username token module

The Username token module is called UsernameTokenSTSModule.

Support for the Username token means that the trust service can create or use tokens in a WebSphere Web services environment. In that WebSphere Web services environment, the Username token passes user identity information in the header of a Web services request.

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Validate
- Issue
- Exchange

Configuration properties (Validate mode)

Skip password validation
Do not perform password validation for the Username token. The default is cleared.

Options for validating the password
Select one of the following options for validating the password for the Username token:

Use Tivoli Access Manager for authentication
Select this default option to use Tivoli Access Manager to authenticate the user.

Use WebSphere Registry for authentication
Select this option to use the WebSphere Application Server registry to authenticate the user.

Use JAAS for authentication
Use Java Authentication and Authorization Service to authenticate the user.

JAAS Login Context Alias
Specifies an alias for the Java Authentication and Authorization Service (JAAS) login context. This field is required. Default: WSLogin

JAAS Provider host name
(Optional) Specifies the host name for the Java Authentication and Authorization Service (JAAS) provider. The default value is localhost.

The system uses the value that you specify in this field. Otherwise, if you have previously defined the host name in a custom properties file, that host name is used. The system uses the default localhost value if you do not specify a host name in this field, and you have not defined it in a custom properties file.

JAAS Provider port
(Optional) Specifies the port for the Java Authentication and Authorization Service (JAAS) provider. The default value is 2809.

The system uses the value that you specify in this field. Otherwise, if you have previously defined the port in a custom properties file, that port is used. The system uses the default value, 2809 if you do not specify a port in this field, and you have not defined it in a custom properties file.

Enable time validity check (based on created time and the amount of time permitted after issue)
Specifies a required created time element on the Username token when checked (default). The software compares the value of the created time element against the value that specifies the amount of time the token is valid after issue.

Amount of time the token is valid after being issued (seconds)

Default: 300 seconds

A value of -1 means that the token does not expire.

Configuration properties (Issue or Exchange mode)

Include nonce in token
Includes a nonce (random bits used for obfuscating the element) in the token. When the password option "4" is specified, this value has no effect.

Include token creation time in token
Adds a timestamp to the token, indicating the creation time of the token.

Options for including password in the token
Indicates whether to include the password in the token. When the password is included, you can specify the format.

Do not include the password
Specifies that you do not want to include the password in the token.

Include the digest of the password value
Specifies that you want to include the password in the token as the digest of the password value.

Include the password in clear text
Specifies that you want to include the password in the token as clear text.

X.509 module

The X.509 module is called X509STSModule.

Validates X.509 security tokens with a token type of:

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-
  wss-x509-token-profile-1.0#X509
http://docs.oasis-open.org/wss/2004/01/oasis-200401-
  wss-x509-token-profile-1.0#X509v3
http://docs.oasis-open.org/wss/2004/01/oasis-200401-
  wss-x509-token-profile-1.0#X509PKIPathv1
```

The module uses the IBM Tivoli Federated Identity Manager KESS to validate the X.509 certificate path.

Deployment scenarios for this module type

- Web services security management
- Custom trust chains

Supported modes

- Validate

Configuration properties

Enable X.509 certificate validation
Specifies whether validation of X.509 certificates must be enforced. By default, this check box is selected. When this box is cleared, the certificate is not validated. This option can be used in deployments where the certificate has already been validated by another entity.

X.509 validator identifier or class name
Specifies a class name for a Tivoli Federated Identity Manager X.509 validator. Leave blank to use the default validator. A custom Tivoli Federated Identity Manager X.509 validator must implement the interface com.tivoli.am.fim.kess.CertificateValidator.

X.509 default value type
If an X.509 BinarySecurityToken does not have the ValueType attribute specified, this configuration value is used as the default ValueType.

Include Subject DN
If enabled, the X.509 Subject Distinguished Name is added to the STSUniversalUser AttributeList.

Include Issuer DN
If enabled, the X.509 Issuer distinguished name is added to the STSUniversalUser AttributeList.

Include Not Before
If enabled, the X.509 NotBefore date is added to the STSUniversalUser AttributeList. This date indicates the earliest date from which the X.509 is valid.

Include Not After
If enabled, the X.509 NotAfter date is added to the STSUniversalUser AttributeList. This date indicates the latest date for which the X.509 is valid.

Include Serial Number
If enabled, the X.509 serial number is added to the STSUniversalUser AttributeList.

Include Type
If enabled, the X.509 type is added to the STSUniversalUser AttributeList.

Include Version
If enabled, the X.509 version is added to the STSUniversalUser AttributeList.

Include Basic Constraints
If enabled, the X.509 Basic Constraints are added to the STSUniversalUser AttributeList.

Please enter a list of Object Identifiers to read from the certificate
Use this text area to add custom Object Identifiers to the STSUniversalUser AttributeList. Put each unique OID on a new line in the text area. Each value is a hexadecimal representation of the octet string.

Token module response files

Before creating a chain mapping using the **manageItfimStsChainMapping** command, you must create a response file. Then, edit the response file so that it contains the appropriate values for your environment.

You can create a response file for a new chain mapping based on an existing chain, or from an existing chain mapping. For each of these operations, you must first run a specific command to create the response file. The content of this created response file depends on the chain for which the chain mapping is based. For more information about creating the response file, see "manageItfimStsChainMapping" on page 304.

Note: If you created a response file that is based on an existing chain mapping, values are automatically specified for many of the parameters shown here.

Available response files for STS chain mapping include:

- "Default mapping module response file" on page 71

Default mapping module response file

The following table provides the parameters, values, and descriptions for the Default mapping module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 1. Parameters in Default mapping module response file (Map mode)

Parameter	Value	Description
self.xslt.rule	*mapping rule*	Specifies the mapping rule to use. Depending on the `self.map.rule.type` value, this rule conforms to either Javascript syntax or XSLT syntax.
self.map.rule.type	*XSLT* or *Javascript*	Specifies whether to use an XSLT or Javascript mapping rule.

Kerberos module response file

The following table provides the parameters, values, and descriptions for the Kerberos module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 2. Parameters in Kerberos module response file

Parameter	Mode	Value	Description
self.KBKeytabFile	Validate	*string*	Specifies the optional path of the Kerberos service keytab file for the Kerberos service identified by the Kerberos security token. If this property is not specified, it must be specified as an STS universal user context attribute.
self.KBRealm	Issue	*string*	Specifies the optional name of the Kerberos realm. The Kerberos realm name must match with a realm configured within krb5.conf. For example, EXAMPLE.COM. If this property is not specified, it must be specified as an STS universal user context attribute.

Table 2. Parameters in Kerberos module response file (continued)

Parameter	Mode	Value	Description
self.KBServiceName	Issue	*string*	Specifies the principal name of the Kerberos service in the form of service-name or service-name@realm. For example, `krb5service/krb5host.example.com` or `krb5service@example.comkrb5service/krb5host.example` When the realm is not supplied, the default realm, if any, configured in krb5.conf is used. The name must match the configuration in Active Directory or whatever Kerberos directory is being used.
self.KBValueType	Issue	*string*	Specifies the value type of the Kerberos binary security token to issue. The default value can be one of the following: • `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ` • `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`

Kerberos delegation module response file

The following table provides the parameters, values, and descriptions for the Kerberos delegation module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 3. Parameters in Kerberos Delegation module response file (Exchange mode)

Parameter	Value	Description
KBDelCredentialCacheSize	*integer*	Determines the number of impersonation handles and user credentials cached for performance reasons in the dynamically loaded library loaded by the module. Set this number to the approximate number of expected concurrent end users of the service for high-volume transactions. The higher the number, the more memory that might be consumed by the IBM Tivoli Federated Identity Manager runtime application. Default: 100
partner.KBDelServicePrincipalName	*string*	(Optional) Specifies the default target Service Principal Name. It is used for WS-Trust clients that do not send the target Service Principal Name in the `AppliesTo ServiceName` element of the RST, and that do not have a mapping rule to configure the target Service Principal Name as an `STSUniversalUser` context attribute.

Table 3. Parameters in Kerberos Delegation module response file (Exchange mode) (continued)

Parameter	Value	Description
partner.KBDelUserSuffix	*DNS*, *CONFIG* or *NONE*	Specifies 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 include: • Do not add a suffix to the username (value: NONE) This option retains the user name. • Add the machine DNS domain as a suffix to the username (value: DNS) This option auto-appends the DNS domain suffix for the IBM 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 for e-business 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:** 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 for e-business user name contains the @ character, this setting has no effect. • Add the configured suffix to the username (value: CONFIG) This option is used to optimize the module behavior for use in Tivoli Access Manager for e-business 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 IBM Tivoli Federated Identity Manager Runtime. This option has no effect when the principal name already contains an @ character.
partner.KBDelConfiguredUserSuffix	*String*	Specifies the suffix to add if using a configured suffix. For example: `@mydomain.com`

KESS STS module response file

The following table provides the parameters, values, and descriptions for the Key Encryption and Signature Service (KESS) STS module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 4. Parameters in KESS STS module response file (Map mode)

Parameter	Value	Description
partner.KessChooseOperations	*Sign, Validate, Encrypt* or *Decrypt*	Specifies the configured operation for the module: sign, validate, encrypt, or decrypt.
partner.KessUserAttribute	*string*	Specifies on which attribute in the STSUU `AttributeList` to perform the operation. This property is common to all operations.
partner.KessSigningKey	*name of keystore and key*	Determines the name of the signing key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).
partner.IncludePublicKey	*true* or *false*	Specifies whether to include the public key in the `KeyInfo`.
partner.IncludeX509IssuerDetails	*true* or *false*	Specifies whether to include issuer information in the `KeyInfo/X509Data`.
partner.IncludeX509SubjectName	*true* or *false*	Specifies whether to include the `X509SubjectName` (distinguished name of signing certificate) in the `KeyInfo/X509Data`.
partner.IncludeX509SubjectKeyIdentifier	*true* or *false*	Specifies whether to include the Subject Key Identifier in the `KeyInfo/X509Data`.
partner.IncludeX509CertificateData	*true* or *false*	Specifies whether to include the `X509CertificateData` in the `KeyInfo/X509Data`.
partner.KessUseInclusiveNamespaces	*true* or *false*	Specifies whether to use the `InclusiveNamespaces` construct, which means employing exclusive XML canonicalization for greater standardization.
partner.sts.kess.verify.signatures	*true* or *false*	Specifies whether to enable signature validation.
partner.sts.kess.validationkey	*name of keystore and key* or *the Subject DN expression for the allowable X.509 certificates*	Determines the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`). You can also use `KeyInfo` of the XML signature to find the X.509 certificate for signature validation. To proceed, enter a regular expression to validate the Subject distinguished name returned in the `KeyInfo`.

Table 4. Parameters in KESS STS module response file (Map mode) (continued)

Parameter	Value	Description
partner.KessValidateWithIncludedCert	*true* or *false*	Specifies whether to use embedded certificate data or a KESS keystore certificate to validate the signature. For example, the KESS keystore might receive a message signed by a key that is not in the KESS keystore. However, the signer of that key is in the keystore. Signature validation might then occur with the certificate embedded in the message.
partner.KessEncryptionKeyIdentifier	*name of keystore and key*	Determines the name of the encryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).
partner.KessEncryptionAlgorithm	*encryption algorithm*	Specifies the block encryption algorithm. Select the identifier from one of the following block encryption algorithms: • Triple DES • AES 128 • AES 192 • AES 256
partner.KessKeyTransportAlgorithm	*transport algorithm*	Specifies the key transport algorithm. Select the identifier from one of the following key transport algorithms: • RSA-1-5 • RSA-OAEP-MGF1P • SHA 1 **Note:** SHA 1 does not come preconfigured. To support this algorithm, you must configure other security providers.
partner.KessDecryptionKeyIdentifier	*name of keystore and key*	Determines the name of the decryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).

LTPA module response file

The following table provides the parameters, values, and descriptions for the Lightweight Third Party Authentication (LTPA) module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 5. Parameters in LTPA module response file

Parameter	Mode	Value	Description
partner.ltpa.filecontents	Validate	*LTPA file contents*	The file contents are extracted from the file exported by the LTPA key export panel of the WebSphere Application Server.
partner.ltpa.password	Validate	*password*	Specifies the password that was used to protect the keys created by the partner. It must be the same password that was used when the keys were created by the partner.
partner.ltpa.usefips	Validate	*true* or *false*	Specifies whether to enable the Federal Information Processing Standards (FIPS). If FIPS was enabled when you created your partner, specify true. The default is false.
self.ltpa.filecontents	Issue	*LTPA file contents*	The file contents are extracted from the file exported by the LTPA key export panel of the WebSphere Application Server.
self.ltpa.password	Issue	*password*	Specifies the password that was used to protect the keys created by the partner. It must be the same password that was used when the keys were created by the partner. This password is required.
self.ltpa.usefips	Issue	*true* or *false*	Specifies whether to enable the Federal Information Processing Standards (FIPS). If FIPS was enabled when you created your partner, specify true. The default is false.
self.ltpa.expiration	Issue	*number of minutes*	Indicates how long, from the time of token creation, the token remains valid. Specify this required value in minutes. You can override this value using the expiration Principle value in the Universal User. The default value is 120 minutes.
self.ltpa.realm	Issue	*string*	Specifies the realm name to append to the user ID during token creation. You can override this value using the realm Principle value in the Universal User. If you do not specify a name here, then it is assumed that the `PrincipleName` in the Universal User document is already in the following form: *`realm:port/userID`*

Table 5. Parameters in LTPA module response file (continued)

Parameter	Mode	Value	Description
self.ltpa.version	Issue	*version number*	Specifies the version number of the LTPA token you are issuing. Specify 1 or 2, denoting LTPA Version 1 or Version 2.
self.ltpa.attributes	Issue	*string*	Specifies the types of attributes to include in the assertion. Use this field only for LTPA Version 2 tokens. An asterisk (*), which is the default setting, includes all of the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, add an attribute under a new `<void method="add">` element.

PassTicket module response file

The following table provides the parameters, values, and descriptions for the PassTicket module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 6. Parameters in PassTicket module response file

Parameter	Mode	Value	Description
partner.PTKApplicationID	Validate	*string*	Specifies the name of the application that was used to generate the unique PassTicket. The name must be no longer than 8 alphanumeric characters. For example: `G1SGRAM`
partner.PTKDESKey	Validate	*hexadecimal digits*	Specifies an optional key value consisting of exactly 16 hexadecimal digits (letters `a` through `f`; integers `0` through `9`) for the PassTicket.
partner.PTKTokenLifetime	Validate	*number of seconds*	Specifies an integer value indicating the amount of time, in seconds, that the token remains valid. There are no minimum or maximum values imposed. Default value: `300` The special value -1 means that the token does not expire.
partner.passticket.verify.signatures	Validate	*true* or *false*	Enables (true) or disables (false) validation of signatures in the token module.
partner.passticket.keystore.alias	Validate	*name of keystore and key*	Determines the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).
self.PTKIncludeNonce	Issue	*true* or *false*	Includes a nonce (random bits used for obfuscating the element) in the token.

Table 6. Parameters in PassTicket module response file (continued)

Parameter	Mode	Value	Description
self.PTKIncludeCreationTime	Issue	*true* or *false*	Adds a timestamp to the token, indicating the creation time of the token.
partner.PTKApplicationID	Issue	*string*	Specifies the name of the application that was used to generate the unique PassTicket. The name must be no longer than 8 alphanumeric characters. For example: `G1SGRAM`
partner.PTKDESKey	Issue	*hexadecimal digits*	Specifies an optional key value consisting of exactly 16 hexadecimal digits (letters `a` through `f`; integers `0` through `9`) for the PassTicket. .
partner.passticket.token.sign	Issue	*true* or *false*	Enables (true) or disables (false) the signing of the PassTicket token module.
partner.PTKSigningKeyIdentifier	Issue	*name of keystore and key*	Determines the name of the signing key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).

SAML 1.0 token module response file

The following table provides the parameters, values, and descriptions for the SAML 1.0 token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 7. Parameters in SAML 1.0 token module response file

Parameter	Mode	Value	Description
self.SAML10OneTimeAssertion Enforcement	Validate	*true* or *false*	Specifies whether this artifact (token) is to be used only once.
partner.SAML10ValidationKey	Validate	*name of keystore and key* or *subject distinguished name expression for the allowable X.509 certificates*	Determines the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`). You can also use `KeyInfo` of the XML signature to find the X.509 certificate for signature validation. To proceed, enter a regular expression to validate the subject distinguished name returned in the `KeyInfo`.
partner.com.tivoli.am.fim.sts.saml.1.0. assertion.verify.signatures	Validate	*true* or *false*	Enables or disables validation of signatures in the token module. Specify true to enable signature validation.
partner.SAML10CreateMultiple UniversalUserAttributes	Validate	*true* or *false*	Specifies whether to keep multiple attribute statements in the groups in which they were received. 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 you specify false (default), multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. This setting is appropriate for most configurations.

Table 7. Parameters in SAML 1.0 token module response file (continued)

Parameter	Mode	Value	Description
self.SAML10AssertionIssuerName	Issue	*name of organization*	Shows a string specifying the name of the organization (for example, a company) that issues the SAML assertions.
self.SAML10AssertionValidBefore	Issue	*number of seconds*	Specifies the 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, but you must specify a value.
self.SAML10AssertionValidAfter	Issue	*number of seconds*	Specifies the amount of time the assertion is valid after being issued (seconds). Default: 60 seconds There is no minimum or maximum value enforced, but you must specify a value.
partner.com.tivoli.am.fim.sts.saml.1.0.assertion.sign	Issue	*true* or *false*	Specifies whether SAML assertions must be signed.
partner.SAML10SigningKeyIdentifier	Issue	*name of keystore and key*	Specifies the name of the signing key identifier and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`
partner.IncludeX509SubjectKeyIdentifier	Issue	*true* or *false*	Specifies whether to include the X.509 subject key identifier with your signature. **No**, the default, excludes the subject key identifier. **Yes** includes it.
partner.IncludePublicKey	Issue	*true* or *false*	Specifies whether to include the public key with your signature. **No**, the default, excludes the public key. **Yes** includes it.
partner.IncludeX509IssuerDetails	Issue	*true* or *false*	Specifies whether to include the issuer name and the certificate serial number with your signature. **No**, the default, excludes the X.509 subject issuer details. **Yes** includes it.
partner.IncludeX509SubjectName	Issue	*true* or *false*	Specifies whether to include the subject name with your signature. **No**, the default, excludes the X.509 subject name. **Yes** includes it.
partner.IncludeX509CertificateData	Issue	*true* or *false*	Specifies whether to include the BASE64 encoded certificate data with your signature. **Yes**, the default, includes the X.509 certificate data. **No** excludes it.

partner.SAML10AssertionSignatureAlgorithm	Issue	**For DSA-SHA1**
		
		`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
		
		For RSA-SHA1
		
		`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
		
		For RSA-SHA256
		
		`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
			Setting that specifies the signature algorithm that is used to sign the SAML assertions for the partner.

Table 7. Parameters in SAML 1.0 token module response file (continued)

Parameter	Mode	Value	Description
partner.SAML10ExtendedAttribute Types	Issue	*attributes*	Specifies the types of attributes to include in the assertion. The default asterisk (*) includes all the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, enter each attribute type. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:1.0:assertion` in the assertion, enter `urn:oasis:names:tc:SAML:1.0:assertion`.
partner.SAML10SubjectConfirmation Method	Issue	*subject confirmation method for assertion*	Specifies the subject confirmation method for the assertion. If you select the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. Valid values can be: • no subject confirmation method (blank) • urn:oasis:names:tc:SAML:1.0:bearer • urn:oasis:names:tc:SAML:1.0:holder-of-key • urn:oasis:names:tc:SAML:1.0:sender-vouches

SAML 1.1 token module response file

The following table provides the parameters, values, and descriptions for the SAML 1.1 token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 8. Parameters in SAML 1.1 token module response file

Parameter	Mode	Value	Description
self.SAML11OneTimeAssertion Enforcement	Validate	*true* or *false*	Specifies whether this artifact (token) is used only once.
partner.SAML11ValidationKey	Validate	*name of keystore and key* or *subject distinguished name expression for the allowable X.509 certificates*	Determines the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`). You can also use `KeyInfo` of the XML signature to find the X.509 certificate for signature validation. To proceed, enter a regular expression to validate the subject distinguished name returned in the `KeyInfo`.
partner.com.tivoli.am.fim.sts.saml.1.1. assertion.verify.signatures	Validate	*true* or *false*	Enables or disables validation of signatures in the token module. Specify true to enable signature validation.
partner.SAML11CreateMultipleUniversal UserAttributes	Validate	*true* or *false*	Specifies whether to keep multiple attribute statements in the groups in which they were received. 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 you specify false (default), multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. This setting is appropriate for most configurations.

Table 8. Parameters in SAML 1.1 token module response file (continued)

Parameter	Mode	Value	Description
self.SAML11AssertionIssuerName	Issue	*name of organization*	Specifies the string for the name of the organization (for example, a company) that issues the SAML assertions.
self.SAML11AssertionValidBefore	Issue	*number of seconds*	Specifies the 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, but you must specify a value.
self.SAML11AssertionValidAfter	Issue	*number of seconds*	Specifies the amount of time the assertion is valid after being issued (seconds). Default: 60 seconds There is no minimum or maximum value enforced, but you must specify a value.
partner.com.tivoli.am.fim.sts.saml.1.1. assertion.sign	Issue	*true* or *false*	Specifies whether SAML assertions must be signed.
partner.SAML11SigningKeyIdentifier	Issue	*name of keystore and key*	Specifies the name of the signing key identifier and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`
partner.IncludeX509SubjectKeyIdentifier	Issue	*true* or *false*	Specifies whether to include the X.509 subject key identifier with your signature. **No**, the default, excludes the subject key identifier. **Yes** includes it.
partner.IncludePublicKey	Issue	*true* or *false*	Specifies whether to include the public key with your signature. **No**, the default, excludes the public key. **Yes** includes it.
partner.IncludeX509IssuerDetails	Issue	*true* or *false*	Specifies whether to include the issuer name and the certificate serial number with your signature. **No**, the default, excludes the X.509 subject issuer details. **Yes** includes it.
partner.IncludeX509SubjectName	Issue	*true* or *false*	Specifies whether to include the subject name with your signature. **No**, the default, excludes the X.509 subject name. **Yes** includes it.
partner.IncludeX509CertificateData	Issue	*true* or *false*	Specifies whether to include the BASE64 encoded certificate data with your signature. **Yes**, the default, includes the X.509 certificate data. **No** excludes it.

Table 8. Parameters in SAML 1.1 token module response file (continued)

Parameter	Mode	Value	Description

partner.SAML11AssertionSignatureAlgorithm	Issue	**For DSA-SHA1**
		
		`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
		
		For RSA-SHA1
		
		`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
		
		For RSA-SHA256
		
		`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
			Setting that specifies the signature algorithm that is used to sign the SAML assertions for the partner.

partner.SAML11ExtendedAttributeTypes	Issue	*attributes*	Specifies the types of attributes to include in the assertion. The default asterisk (*) setting includes all the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, enter each attribute type. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:1.0:assertion` in the assertion, enter `urn:oasis:names:tc:SAML:1.0:assertion`.
partner.SAML11SubjectConfirmation Method	Issue	*subject confirmation method for assertion*	Specifies the subject confirmation method for the assertion. If you select the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. Valid values can be: • no subject confirmation method (blank) • urn:oasis:names:tc:SAML:1.0:bearer • urn:oasis:names:tc:SAML:1.0:holder-of-key • urn:oasis:names:tc:SAML:1.0:sender-vouches
partner.SAML11IncludeInclusive Namespaces	Issue	*true* or *false*	Specifies whether to use the `InclusiveNamespaces` construct, which means employing exclusive XML canonicalization for greater standardization. The default is cleared.

SAML 2.0 token module response file

The following table provides the parameters, values, and descriptions for the SAML 2.0 token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 9. Parameters in SAML 2.0 token module response file

Parameter	Mode	Value	Description
self.SAML20OneTimeAssertion Enforcement	Validate	*true* or *false*	Specifies whether this artifact (token) is used only once.
partner.com.tivoli.am.fim.sts.saml.2.0. assertion.verify.signatures	Validate	*true* or *false*	Enables or disables validation of signatures in the token module.
partner.SAML20ValidationKey	Validate	*name of keystore and key*	Determines the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).
partner.SAML20DecryptionKey	Validate	*name of keystore and key*	Specifies the name of the decryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`(for example, `DefaultKeyStore_testkey`).
partner.SAML20CreateMultipleUnivsersal UserAttributes	Validate	*true* or *false*	Specifies whether to keep multiple attribute statements in the groups in which they were received. 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 you specify false (default), multiple attribute statements are arranged into a single group (AttributeList) in the `STSUniversalUser` document. This setting is appropriate for most configurations.
partner.SAML20MapUnknownIdentifiers ToAnonymousUser	Validate	*true* or *false*	Specifies that the service provider can map an unknown persistent name identifier alias to the anonymous user account. The default is false.

partner.com.tivoli.am.fim.sts.saml.2.0. assertion.default.nameidformat	Validate	*NameId format*	Specifies a parameter for use during validation of a SAML assertion. The parameter determines processing rules for the `NameID` element when one of the following conditions exists:
			• If there is not an explicit Format attribute included in the assertion
			• If the Format attribute is
			`urn:oasis:names:tc:SAML:1.1: nameid-format:unspecified`
			
			Typically, this parameter is needed only for STS chains that process SAML assertions that do not set the Format attribute. A normal example value is:
			`urn:oasis:names:tc:SAML:1.1: nameid-format:emailAddress`

self.SAML20Issuer	Issue	*name of organization*	Specifies the string for the name of the organization (for example, a company) that issues the SAML assertions.
self.SAML20AssertionValidBefore	Issue	*number of seconds*	Default: 60 seconds. There is no minimum or maximum value enforced, but a value is required.
self.SAML20AssertionValidAfter	Issue	*number seconds*	Default: 60 seconds. There is no minimum or maximum value enforced, but a value is required.

Table 9. Parameters in SAML 2.0 token module response file (continued)

Parameter	Mode	Value	Description
partner.SAML20ExtendedAttributeTypes	Issue	*attribute types*	Specifies the types of attributes to include in the assertion. The asterisk (*), which is the default setting, includes all the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, enter each attribute type. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:2.0:assertion` in the assertion, enter `urn:oasis:names:tc:SAML:2.0:assertion`
partner.com.tivoli.am.fim.sts.saml.2.0. assertion.sign	Issue	*true* or *false*	Specifies whether SAML assertions must be signed.
partner.SAML20SigningKeyIdentifier	Issue	*name of keystore and key*	Specifies the name of the signing key identifier and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
partner.IncludeX509CertificateData	Issue	*true* or *false*	Specifies whether to include the BASE64 encoded certificate data with your signature. **Yes**, the default, includes the X.509 certificate data. **No** excludes it.
partner.IncludeX509SubjectName	Issue	*true* or *false*	Specifies whether to include the subject name with your signature. **No**, the default, excludes the X.509 subject name. **Yes** includes it.
partner.IncludeX509SubjectKeyIdentifier	Issue	*true* or *false*	Specifies whether to include the X.509 subject key identifier with your signature. **No**, the default, excludes the subject key identifier. **Yes** includes it.
partner.IncludeX509IssuerDetails	Issue	*true* or *false*	Specifies whether to include the issuer name and the certificate serial number with your signature. **No**, the default, excludes the X.509 subject issuer details. **Yes** includes it.
partner.IncludePublicKey	Issue	*true* or *false*	Specifies whether to include the public key with your signature. **No**, the default, excludes the public key. **Yes** includes it.

Table 9. Parameters in SAML 2.0 token module response file (continued)

Parameter	Mode	Value	Description

partner.SAML20SignatureAlgorithm	Issue	**For DSA-SHA1**
		
		`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
		
		For RSA-SHA1
		
		`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
		
		For RSA-SHA256
		
		`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
			Setting that specifies the signature algorithm that is used to sign the SAML assertions for the partner.

partner.SAML20EncryptionKey	Issue	*name of keystore and key*	Specifies the name of the encryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname` (for example, `DefaultKeyStore_testkey`).
partner.SAML20EncryptAssertions	Issue	*true* or *false*	Specifies whether assertions are to be encrypted. If selected, you must specify an encryption key.
partner.SAML20EncryptAttributes	Issue	*true* or *false*	Specifies whether Attribute elements within the assertions are to be encrypted. If selected, you must specify an encryption key.
partner.SAML20EncryptNameID	Issue	*true* or *false*	Specifies whether `NameID` elements in the assertions are to be encrypted. If selected, you must specify an encryption key.
partner.SAML20EncryptionAlgorithm	Issue	*encryption algorithm*	Specifies the identifier for the encryption algorithm to use to encrypt data for this partner: • Triple DES (Triple Digital Encryption Standard) • AES-128 (Advanced Encryption Standard 128-bit) • AES-192 (Advanced Encryption Standard 192-bit) • AES-256 (Advanced Encryption Standard 256-bit)

Table 9. Parameters in SAML 2.0 token module response file (continued)

Parameter	Mode	Value	Description
partner.SAML20SubjectConfirmation Method	Issue	*subject confirmation method for assertion*	Specifies the subject confirmation method for the assertion. You can add one or more subject confirmation methods at the same time, or choose not to specify a confirmation method. If you specify the holder-of-key type, the default includes the X.509 Certificate Data in the `KeyInfo` for the `SubjectConfirmationMethod`. Valid values can be: • no subject confirmation method (blank) • urn:oasis:names:tc:SAML:2.0:bearer • urn:oasis:names:tc:SAML:2.0:holder-of-key • urn:oasis:names:tc:SAML:2.0:sender-vouches

IBM Tivoli Access Manager for e-business authorization module response file

The following table provides the parameters, values, and descriptions for the Tivoli Access Manager for e-business authorization module response file. Edit the response file to ensure that you have the appropriate values for your environment.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Table 10. Parameters in Tivoli Access Manager for e-business authorization module response file (Authorize mode)

Parameter	Value	Description
TAMAuthzHaltIfUnauthorized	*true* or *false*	Stops the trust service module chain execution if the user is not authorized. If the user is unauthorized and the value is set to false, then the trust service chain process continues. If the user is authorized, then this parameter is ignored.

IBM Tivoli Access Manager for e-business credential module response file

The following table provides the parameters, values, and descriptions for the Tivoli Access Manager for e-business credential token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Table 11. Parameters in Tivoli Access Manager for e-business credential module response file

Parameter	Mode	Value	Description
partner.IVEnableSignatureValidation	Validate	*name of keystore and key*	Specifies the name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`. For example, `DefaultKeyStore_testkey`.
partner.ivcred.verify.signatures	Validate	*true* or *false*	Specifies the validation key that the partner must use. Enables (true) or disables (false) validation of signatures in the token module.
self.IVExtendedAttributeTypes	Issue	*assertion attributes*	Specifies the types of attributes to include in the assertion. The default asterisk (*) includes all the attribute types that are specified in the identity mapping file or by the custom mapping module. To specify one or more attribute types individually, enter each attribute type. For example, if you want to include only attributes of type urn:ibm:names:ITFIM:5.1:accessmanager in the assertion, enter `urn:ibm:names:ITFIM:5.1:accessmanager`.
self.IVEnableSignatures	Issue	*name of keystore and key*	Specifies the name of the signing key identifier and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`. For example, `DefaultKeyStore_testkey`.
self.IncludePublicKey	Issue	*true* or *false*	Specifies whether to include this attribute. **No**, the default, excludes the public key. If **Yes**, the public key of the signing certificate is included in the Base64 encoded form. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_PUBKEY`.
self.IncludeX509SubjectName	Issue	*true* or *false*	Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Name. If **Yes**, the distinguished name of the subject for the signing certificate is included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_SUBJECT`.
self.IncludeX509IssuerDetails	Issue	*true* or *false*	Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Issuer Details. If **Yes**, the issuer details of the signing certificate are included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_ISSUER`.
self.IncludeX509SubjectKeyIdentifier	Issue	*true* or *false*	Specifies whether to include this attribute. **No**, the default, excludes the X509 Subject Key Identifier. If **Yes**, the subject key identifier of the signing certificate is included. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE_SKI`.
self.IncludeX509CertificateData	Issue	*true* or *false*	Specifies whether to include this attribute. **No**, the default, excludes the X509 Certificate Data. If **Yes**, the certificate data of the signing certificate is included in the Base64 encoded form. The extended attribute is labeled `ITFIM_IVCRED_SIGNER_CERTIFICATE`.
self.ivcred.add.signatures	Issue	*true* or *false*	Specifies that signatures must be added to tokens.

Tivoli Directory Integrator module response file

The following table provides the parameters, values, and descriptions for the Tivoli Directory Integrator module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 12. Parameters in Tivoli Directory Integrator module response file (Map mode)

Parameter	Value	Description
self.tdi.adaptor	*string*	Specifies the format to name the Work Entry attributes (for example, `name` or `name_type`). The default value is `name`. Use the `name_type` format if multiple attributes with the same name exist.
self.tdi.hostname	*string*	Specifies the 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`
self.tdi.port	*integer*	Specifies the port number on which the Tivoli Directory Integrator server is configured to run. The default value is 1099.
self.tdi.poolsize	*integer*	Specifies the number of assembly line handlers to maintain for this trust chain. The value must be a positive integer. The default value is 10.
self.tdi.maxwaitingthreads	*integer*	Specifies the 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.
self.tdi.maxwaittime	*-1*, *0*, or *a positive integer representing milliseconds*	Determines the amount of time for threads to wait for an assembly line handler to become available. **Wait indefinitely (value: -1)** 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 (value: 0)** 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. **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.
self.tdi.config	*file name*	Specifies the solution name, or the file name of the configuration file, to use (for example, `tdi_demo_mappings.xml`). This parameter is required, along with the **self.tdi.assemblyline** parameter value, for manually configuring the settings.

Table 12. Parameters in Tivoli Directory Integrator module response file (Map mode) (continued)

Parameter	Value	Description
self.tdi.assemblyline	*file name*	Specifies the name of the assembly line to use (for example, `assemblyLine1`). This parameter is required, along with the **self.tdi.config** parameter value, for manually configuring the settings.
partner.tdi.adaptor	*string*	Specifies the format to name the Work Entry attributes (for example, `name` or `name_type`). The default value is `name`. Use the `name_type` format if multiple attributes with the same name exist.
partner.tdi.hostname	*string*	Specifies the 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`
partner.tdi.port	*integer*	Specifies the port number on which the Tivoli Directory Integrator server is configured to run. The default value is 1099.
partner.tdi.poolsize	*integer*	Specifies the number of assembly line handlers to maintain for this trust chain. The value must be a positive integer. The default value is 10.
partner.tdi.maxwaitingthreads	*integer*	Specifies the 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.
partner.tdi.maxwaittime	*-1*, *0*, or *a positive integer representing milliseconds*	Determines the amount of time for threads to wait for an assembly line handler to become available. **Wait indefinitely (value: -1)** Do not put a limit on the time for threads to wait for the assembly line handler to become available. This is the default option. **Do not wait for assembly line handler after initial try (value: 0)** 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. **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.

Table 12. Parameters in Tivoli Directory Integrator module response file (Map mode) (continued)

Parameter	Value	Description
partner.tdi.config	*file name*	Specifies the solution name, or the file name of the configuration file, to use (for example, `tdi_demo_mappings.xml`). This parameter is required, along with the **self.tdi.assemblyline** parameter value, for manually configuring the settings.
partner.tdi.assemblyline	*file name*	Specifies the name of the assembly line to use (for example, `assemblyLine1`). This parameter is required, along with the **self.tdi.config** parameter value, for manually configuring the settings.

Username token module response file

The following table provides the parameters, values, and descriptions for the Username token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 13. Parameters in Username token module response file

Parameter	Mode	Value	Description
self.UTSkipPasswordValidation	Validate	*true* or *false*	Specifies whether to perform password validation for the Username token. The default is false.
self.UTPasswordValidationOptions	Validate	*TAM*, *WAS*, or *JAAS*	Specifies one of the following options for validating the password for the Username token: **Use Tivoli Access Manager for authentication** Specify this default option to use Tivoli Access Manager for e-business to authenticate the user. **Use WebSphere Registry for authentication** Specify this option to use the WebSphere Application Server registry to authenticate the user. **Use JAAS for authentication** Specify Java Authentication and Authorization Service to authenticate the user.
self.UTJAASModuleAlias	Validate	*alias*	Specifies an alias for the Java Authentication and Authorization Service (JAAS) login context. Default: `WSLogin`

Table 13. Parameters in Username token module response file (continued)

Parameter	Mode	Value	Description
self.UTJAASHostName	Validate	*JAAS host name*	Specifies the optional host name for the Java Authentication and Authorization Service (JAAS) provider. The default value is `localhost`. If you specify a value here, it is used. Otherwise, if you have previously defined the host name in a custom properties file, that host name is used. If you do not specify a host name, and you have not defined it in a custom properties file, the default of `localhost` is used.
self.UTJAASPort	Validate	*port for JAAS provider*	Specifies the optional port for the Java Authentication and Authorization Service (JAAS) provider. The default value is `2809`. If you specify a value here, it is used. Otherwise, if you have previously defined the port in a custom properties file, that port is used. If you do not specify a port, and you have not defined it in a custom properties file, the default of `2809` is used.
partner.UTValidateFreshness	Validate	*true* or *false*	Specifies a created time element on the Username token when checked (default). The software compares the value of the created time element against the value that specifies the amount of time the token is valid after issue.
partner.UTFreshnessLimit	Validate	*number of seconds*	Specifies, in seconds, the amount of time the Username token is valid after being issued. Default: 300 seconds A value of -1 means that the token does not expire.
self.UTIncludeNonce	Issue	*true* or *false*	Includes a nonce (random bits used for obfuscating the element) in the token. When you specify **Do not include the password option**, this value is ineffective.
self.UTIncludeCreationTime	Issue	*true* or *false*	Adds a timestamp to the token, indicating the creation time of the token.

Table 13. Parameters in Username token module response file (continued)

Parameter	Mode	Value	Description
partner.UTPasswordOptions	Issue	*password options*	Indicates whether to include the password in the token. When the password is included, you can specify the format with the following options: **4 - Do not include the password** Specifies that you do not want to include the password in the token. **2 - Include the digest of the password value** Specifies that you want to include the password in the token as the digest of the password value. **3 - Include the password in clear text** Specifies that you want to include the password in the token as clear text.

X509 token module response file

The following table provides the parameters, values, and descriptions for the X509 token module response file. Edit the response file to ensure that you have the appropriate values for your environment.

Table 14. Parameters in X509 token module response file (Validate mode)

Parameter	Value	Description
partner.X509EnableValidation	*true* or *false*	Specifies whether validation of X.509 certificates must be enforced (default). When not specified, the certificate is not validated. This option can be used in deployments where the certificate has already been validated by another entity.
partner.X509ValidatorID	*validator*	Specifies a class name for a Tivoli Federated Identity Manager X.509 validator. Leave blank to use the default validator. A custom Tivoli Federated Identity Manager X.509 validator must implement the interface com.tivoli.am.fim.kess.CertificateValidator.
partner.X509DefaultValueType	*configuration value*	If an X.509 `BinarySecurityToken` does not have the `ValueType` attribute specified, this configuration value is used as the default `ValueType`. The configuration value can be one of the following: • empty value (blank) • http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3 • http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509 • http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1
partner.X509IncludeSubjectDN	*true* or *false*	If enabled, the X.509 Subject Distinguished Name is added to the `STSUniversalUser AttributeList`.

Table 14. Parameters in X509 token module response file (Validate mode) (continued)

Parameter	Value	Description
partner.X509IncludeIssuerDN	*true* or *false*	If enabled, the X.509 Issuer Distinguished Name is added to the `STSUniversalUser AttributeList`.
partner.X509IncludeNotBefore	*true* or *false*	If enabled, the X.509 `NotBefore` date is added to the `STSUniversalUser AttributeList`. This date indicates the earliest date from which the X.509 is valid.
partner.X509IncludeNotAfter	*true* or *false*	If enabled, the X.509 `NotAfter` date is added to the `STSUniversalUser AttributeList`. This date indicates the latest date for which the X.509 is valid.
partner.X509IncludeSerialNumber	*true* or *false*	If enabled, the X.509 serial number is added to the `STSUniversalUser AttributeList`.
partner.X509IncludeType	*true* or *false*	If enabled, the X.509 type is added to the `STSUniversalUser AttributeList`.
partner.X509IncludeVersion	*true* or *false*	If enabled, the X.509 version is added to the `STSUniversalUser AttributeList`.
partner.X509IncludeBasicConstraints	*true* or *false*	If enabled, the X.509 Basic Constraints are added to the `STSUniversalUser AttributeList`.
partner.X509ObjectIdentifiersToRead	*object identifiers*	Adds custom Object Identifiers to the `STSUniversalUser AttributeList`. Each value is a hexadecimal representation of the octet string. Specify each Object Identifier in a separate value element.

Chapter 8. Managing certificates and keystores in the key service

You can perform several operations on a keystore that has already been imported into the IBM Tivoli Federated Identity Manager key service or on specific certificates within an existing keystore.

About this task

There are two types of keystores used in the IBM Tivoli Federated Identity Manager key service:

Keystores
Private keys and personal certificates are stored in *keystores*. These keys and certificates include signing keys, decryption keys, and SSL client certificates (in the keystore of the client).

Truststores
Public keys and CA certificates are stored in *truststores*. These keys and certificates include validation keys, encryption keys, CA certificates for servers on which SSL authentication is configured (in the truststore of the client).

The two types of keystores are collectively referred to as *keystores*. The term *truststores* is used here when an instruction is specifically referring to the keystores in which public keys and CA certificates are stored.

Tasks that manage the keystores:

- "Changing a keystore password" on page 96
- "Deleting a keystore" on page 97

Tasks that manage certificates that have already been imported into a keystore:

- "Viewing certificates in a keystore" on page 96
- "Disabling a certificate" on page 107
- "Modifying certificate settings" on page 107
- "Enabling a certificate" on page 108
- "Deleting a certificate" on page 109
- "Exporting a certificate" on page 109

Tasks related to replacing existing certificates, such as when a certificate expires or the password of a certificate has been compromised:

- "Replacing an existing certificate" on page 97, including:
  - "Obtaining your replacement certificates" on page 98
  - "Obtaining replacement certificates from your partner" on page 103

For information about tasks that are generally used during the initial configuration of your environment or the initial configuration of a federation, refer the *IBM Tivoli Federated Identity Manager Configuration Guide*. These tasks include:

- Preparing keystores, including creating a keystore and importing a keystore
- Planning the types of certificates you will need in your environment, including your certificates and the certificates of your partners

- Obtaining your certificates, including creating self-signed certificates or creating CA-signed certificates
- Importing certificates or receiving signed certificates into a keystore
- Obtaining certificates from your partner
- Updating the cryptography policy
- Modifying the certificate revocation settings
- Reloading the Federated Identity Manager runtime configuration

Viewing certificates in a keystore

An inventory of all the certificates in a keystore helps you manage those keys better. You can use the Integrated Solutions Console to view certificates in the keystore or truststore.

About this task

Use this task to see a list of certificates contained in a keystore or truststore. From this console panel, you can choose other certificate administration tasks.

To view certificates in the keystore or truststore:

Procedure

1. Log on to the console.
2. Cclick **IBM Tivoli Federated Identity Manager > Configure Key Service > Keystores**. The Keystores panel opens.
3. Select a keystore or truststore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Keys panel opens, and prompts for a password.
5. Enter the keystore or truststore password.
6. Click **OK**.

   The password for the DefaultKeyStore is `testonly`.

   This keystore and its password are intended for use when testing or prototyping. Change the password to use this keystore in a production environment. The Keys panel opens. Keys in the selected keystore or truststore are listed.
7. You can complete additional tasks from this panel. Continue with the instructions for the task:
   - "Exporting a certificate" on page 109
   - "Enabling a certificate" on page 108
   - "Deleting a certificate" on page 109
   - "Disabling a certificate" on page 107 or "Enabling a certificate" on page 108
   - "Modifying certificate settings" on page 107
   - "Exporting a certificate" on page 109

     For additional information about importing or receiving a certificate into a keystore or truststore, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Changing a keystore password

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

About this task

You can change the keystore passwords in any of the following situations:

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

Procedure

1. Log on to the console.
2. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**.

   The Keystores panel opens.
3. Select a keystore from the Keystore table. The **Change Password** button is activated.
4. Click **Change Password**. The Change Keystore Password panel opens.
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**.

Deleting a keystore

Use the IBM Tivoli Federated Identity Manager to remove a keystore or truststore you no longer need.

About this task

Deleting the keystores also removes the keys and certificates that are used to secure the content and transport messages.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**. The Keystores panel opens.
3. Select a keystore or truststore.
4. Click **Delete**. A message box asks you to confirm that you want to delete the specified keystore or truststore.
5. Click **OK** to delete the keystore or truststore, or click **Cancel** to exit from the window without deleting the keystore or truststore.

Replacing an existing certificate

Certificates are used to secure the content of messages that are transmitted between partners in a federation. During the initial configuration of your environment, set up the message security for your federation by adding the necessary certificates to your IBM Tivoli Federated Identity Manager key service for you and your partner. You can also set up transport security, which might

include configuring client certificates. There might be times when you must replace one of these certificates, such as when a certificate expires or the security of a certificate has been compromised.

About this task

The steps for replacing an existing certificate depend on whether you are replacing your own certificate or the certificate of your partner.

Replacing one of your signing or decryption certificates involves the following tasks:

1. "Obtaining your replacement certificates," using one of the following methods:
   - "Replacing a self-signed certificate"
   - "Requesting a CA-signed certificate" on page 100 and "Receiving a signed certificate" on page 101.
2. "Disabling a certificate" on page 107.
3. "Modifying certificate settings" on page 107.
4. "Enabling a certificate" on page 108.
5. "Deleting a certificate" on page 109.
6. "Exporting a certificate" on page 109.

Replacing one of your partner's signing, validation, server validation, or encryption certificates involves the following tasks:

1. "Obtaining replacement certificates from your partner" on page 103
2. "Disabling a certificate" on page 107.
3. "Modifying certificate settings" on page 107.
4. "Enabling a certificate" on page 108.
5. "Deleting a certificate" on page 109.

Obtaining your replacement certificates

You have two options for how you should obtain your replacement certificates. The method depends on whether you will use the certificate in a test environment or a production environment.

About this task

- In a test environment, you could use as self-signed certificate. See "Replacing a self-signed certificate."
- In a production environment, you would want to request your keys from a certificate authority. See "Requesting a CA-signed certificate" on page 100.

Replacing a self-signed certificate

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 or private key pair that is randomly generated and is signed by its own private key.

Procedure

1. Create a new self-signed certificate.
   a. Log on to the console.
   b. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service -> Keystores**.

      The Keystores panel opens.
   c. Select a keystore from the Keystore table. The **View Keys** button is activated.
   d. Click **View Keys**. The Password panel opens.
   e. Type your keystore password.
   f. Click **OK**.
   g. Click **Create Self-Signed Certificate**. The Create Self-Signed Certificate panel opens.
   h. Complete the fields.
   i. Then click **OK**. A public/private key pair is added to the keystore.
   j. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button. The self-signed certificate was added to your keystore.
2. (Optional) Make sure that the currently configured certificate is not in use and is not enabled. Follow these steps to verify this setting, perform the following steps:
   a. On the Keystores panel, select a keystore from the Keystore table. The **View Keys** button is activated.
   b. Click **View Keys**. The Password panel opens.
   c. Type your keystore password.
   d. Click **OK**.
   e. Select a key from the keys table.
   f. Click **Disable** to disable the key.
3. Modify your certificate configuration settings.
   a. In the console navigation, click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on**.
   b. Locate the certificate configuration that you want to change. For example, click either **Federations** or **Partners**. The panel shows a list of configured federations or partners.
   c. Select the federation or partner you want to update.
   d. Click **Properties** to view the currently configured properties. The properties shown on this panel are described in the online help.
   e. Locate the sections in which the certificates are defined and change the certificates named in those fields to the names of the certificates you want to use.

      For example, you must change the certificate that is specified in any of the following Properties sections:

      - **Signature Options** or **Signatures**
      - **Encryption**
      - **SOAP SSL Connection Parameters**, (only in the Partner Properties), which include:
        - **Server Validation Certificate**
        - **Client Certificate**

f. When you have finished modifying properties, click **OK** to close the Federation Properties panel.

What to do next

Enable the certificate as described in "Enabling a certificate" on page 108.

Requesting a CA-signed certificate

In a production environment, you must obtain your certificates for signing, decryption, and client authentication from a certificate authority that signs 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.

Note: Because you must use the CA-signed certificate for signing or decryption, or as an SSL client authentication certificate (if you are a client), use a *keystore* and not a truststore in this procedure.

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 uses the data contained within the CSR and generates 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.

Attention: Do not use this procedure to request a certificate to use in your WebSphere Application Server keystores. WebSphere Application Server cannot accept requests that are generated by this procedure, which uses a IBM Tivoli Federated Identity Manager key service utility. If you need a certificate for use in WebSphere Application Server, see the information in Chapter 9, "Managing the SSL configuration," on page 113.

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

Procedure

1. Log on to the console.

2. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**.

   The Keystores panel opens.
3. Select a keystore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel opens.
5. Type your keystore password.
6. Click **OK**.
7. Click **Create Certificate Request**. The Create a certificate request panel opens.
8. Complete the fields.
9. Then click **OK**. The Generated Certificate Signature Request window opens.
10. 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.
11. 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 must be replaced with the signed certificate from the CA.
12. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

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. You might also want to use one certificate for all activities.

When you have created all of your certificate sign requests, follow your CA instructions for transmitting the request file. Then, continue with the steps for receiving a CA certificate from the CA in "Receiving a signed certificate."

Receiving a signed certificate

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 "Creating a certificate request" on page 115 and have saved the request certificate a location that is accessible to the key service.

Note: When you use the CA-signed certificate for signing or decryption, or as an SSL client authentication certificate (if you are a client), you must use a *keystore* and not a truststore in this procedure.

Procedure

1. Log on to the console.
2. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**.

   The Keystores panel opens.

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 opens.
5. Type your keystore password
6. Click **OK**.
7. Click **Receive Certificate from CA**.
8. Select the location of the certificate that you received from the CA. Then click **OK**. The temporary self-signed certificate in the keystore is replaced with the received signed certificate.
9. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

When the certificate is created and added to the keystore, it is enabled by default. Consider disabling it until you have modified your configuration to use it. See "Disabling a certificate" on page 107 for instructions. Then, continue with the steps for modifying your certificate settings, as described in "Modifying certificate settings" on page 107.

Importing a certificate

The method to import a certificate into a keystore depends on how you have obtained the keys.

Before you begin

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

About this task

You must import a certificate if you obtained it in either of the following ways:

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

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

Imported keys are enabled by default.

Procedure

1. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**.

   The Keystores panel opens.

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**.
4. Enter the keystore password when prompted.
5. Click **OK**. The Keys panel opens. Keys in the selected keystore are listed.
6. Click the **Import** button. The Key Wizard starts and opens the Welcome panel.
7. Click **Next**. The Keystore Format panel opens.
8. Select the appropriate **Keystore format** for the file you want to import. 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
9. Then, click **Next**. The **Upload Key File** panel opens.
10. Specify the path to the location of the key, and if prompted, a password for the key file.
11. Click **Next**.
12. Specify a label for the key and, if prompted, select the key to import.
13. Click **Next**. A summary panel opens.
14. Click **Finish** to exit the wizard.
15. Repeat these steps to import all the keys and certificates that you must replace.

Obtaining replacement certificates from your partner

After the initial configuration of the federation properties of your partner, you might find that you replace one or more of the certificates that you use to validate its signatures or to encrypt data that you send to it.

About this task

When you initially add a partner to your federation, you typically add the data and certificates of your partner using a metadata file that the partner provides to you. However, if you have already imported the data of your partner, you can receive a new certificate from your partner (such as over FTP, through e-mail, or another transfer method) and import it, as described in the following procedure.

Use this task to import either:

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

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

Imported keys are enabled by default.

Note: Because the certificates in this procedure are the public keys of your partner that you must use to validate the signature or encrypt data to your partner, you must use a *truststore* and not a keystore in this procedure.

Procedure

1. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**.

   The Keystores panel opens.
2. Select a truststore to store the keys of your partner from the Keystore table. The **View Keys** button is activated.
3. Click **View Keys**.
4. Enter the truststore password.
5. Click **OK**. The Keys panel opens. Keys in the selected keystore are listed.
6. Click the **Import** button. The Key Wizard starts and opens the Welcome panel.
7. Click **Next**. The Keystore Format panel opens.
8. 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

   JKS
   Java Key Store

   The **Upload Key File** panel opens.
9. Specify a label for the key and, if prompted, select the key to import.
10. Then click **Next**. A summary panel opens.
11.
12. Click **Finish** to exit the wizard.
13. Repeat these steps to import all the keys and certificates to use in the federation.
14. Click **Finish** to exit the wizard.

What to do next

When the certificate is imported to the keystore, it is enabled by default.

Disable it until you have modified your configuration to use it. See “Disabling a certificate” on page 107 for instructions.

Then, continue with the steps for modifying the certificate settings for the certificate of your partner, as described in “Modifying certificate settings” on page 107.

Reloading runtime configuration

Reload the IBM Tivoli Federated Identity Manager runtime console when a new key or certificate for DN is added so that configuration changes are recognized.

About this task

When you start the WebSphere Application Server where the IBM Tivoli Federated Identity Manager runtime is installed, the IBM Tivoli Federated Identity Manager runtime reads the keystores. In addition, it builds a map of DN-to-list-of-keys-certificates as a part of the initialization of the IBM Tivoli Federated Identity Manager key service.

When a new key/certificate is added using the IBM Tivoli Federated Identity Manager console, the key/certificate is stored in the specified keystore on the disk.

You must reload the IBM Tivoli Federated Identity Manager runtime configuration to:

- Reread the keystores.
- Rebuild the map.
- Make these new keystores and the new map available for use by the IBM Tivoli Federated Identity Manager key service.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management** to open the Runtime Node Management panel.
3. Click **Reload Configurations**.

Processing of keystore

When the IBM Tivoli Federated Identity Manager key service is initialized, a map of DN-to-list-of-keys-certificates is built from the keys/certificates in the keystores.

Note: Keystores and the keys or certificates in each of these keystores are processed in an unspecified order.
When a key or a certificate for a DN X is processed, it is added to the DN X's list-of-keys-certificates in the map using the following rule:

1. If the key or certificate is not identical to any of the keys and certificates in the list, then the key or certificate is added to the list. The keys and certificates in the list are ordered by their expiration date. The key selection criteria determines whether ascending or descending order is used.
2. If the key or certificate is identical to a key and certificate in the list, then the key or certificate is discarded.
3. If the new certificate's signing key is already stored, then the certificate is discarded.
4. If the new key is a signing key for an already stored certificate for X:
   a. The certificate is discarded.
   b. The key is added to the map.
   c. X's key or certificate mapping is changed to point to this new key.

The keys or certificates are managed in such a way that enables auto key rollover. Communication occurs using both keys and certificates while the new certificate is disseminated to services that must use the certificate.

See the following example on how a keystore is processed assuming that they are processed in the following order:

Table 15. Example keystores to be processed in a specified order

Processing order	Keystore name	List of keys or certificates
1	certstore1 (CA certificates)	`certA1 [DN: CN=A,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 1234]` `certA1 [DN: CN=A,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 1234]` `certA1 [DN: CN=A,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 1234]`

2	keystore1 (signing/ encryption keys)	`keyA1 [DN: CN=A,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 1234]`
		
		`keyB1 [DN: CN=B,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 2345]`

3	certstore2 (CA certificates)	`certC3 [DN: CN=C,O=Comp,C=US; Expires: Dec 31, 2007; Serial: 5678]`
4	keystore2 (signing/ encryption keys)	`keyB2 [DN: CN=B,O=Comp,C=US; Expires: Dec 31, 2010; Serial: 2345]`

The following table shows the resulting map of DN-to-list-of-keys-certificates with an *ascending order* key selection criteria.

Table 16. Resulting map of DN-to-list-of-keys-certificates

DN	List of keys or certificates	Explanation
`DN CN=A,O=Comp,C=US`	`keystore1_keyA1`	This signing/encryption key takes precedence over the certificate `certstore1_certA1`.
`DN CN=B,O=Comp,C=US`	`keystore1_keyB1`	This is the first key that is read. The duplicate key `keystore2_keyB2` is discarded.
`DN CN=C,0=Comp,C=US`	`certstore1_certC1` and `certstore1_certC2`	The duplicate certificate `certstore2_certC3` is discarded.

In WebSphere Application Server version 6.0.2, it cannot store a certificate as a public or private key pair in a keystore stores signing or encryption keys. This is a limitation of the `javax.net.ssl` that is shipped with Java Secure Socket Extension (JSSE) that is used by this WebSphere Application Server version.

The scenario in the example occurs if the specification for a CA certificate was `keystore1_key1` OR `certstore1_certA1`, because the `keystore1_key1` key takes precedence. This is considered an improper configuration. A public certificate must be stored as a public signer certificate in a trusted keystore with CA certificates. Secure communications must not use the same DN if the server signs with a key and a client validates with a certificate.

This limitation causes `SSLHandshakeException` to be thrown when the IBM Tivoli Federated Identity Manager runtime attempts to establish an SSL connection. The stack trace of the exception is similar to the following stack trace:

```
4/19/07 16:26:42:233 GMT] 00000048 HttpClientImp I
com.tivoli.am.fim.soap.client.HttpClientImpl doRequest
  javax.net.ssl.SSLHandshakeException: unknown certificate
```

```
at com.ibm.jsse.bv.a(bv.java:67)
at com.ibm.jsse.bv.startHandshake(bv.java:163)
at com.ibm.net.ssl.www2.protocol.https.b.o(b.java:136)
at com.ibm.net.ssl.www2.protocol.https.i.connect(i.java:28)
at com.ibm.net.ssl.www2.protocol.http.bc.getOutputStream(bc.java:44)
at com.ibm.net.ssl.www2.protocol.https.l.getOutputStream(l.java:23)
at com.tivoli.am.fim.soap.client.HttpClientImpl.sendRequest(Unknown Source)
at com.tivoli.am.fim.soap.client.HttpClientImpl.doRequest(Unknown Source)
...
```

To eliminate this problem, configure the IBM Tivoli Federated Identity Manager runtime to use `IBMJSSE2` as the provider for the Java Secure Socket Extension (JSSE). The Java Secure Socket Extension (JSSE) supports the extraction of a public certificate from a private or public key pair. Use the runtime custom property `com.tivoli.am.fim.soap.client.jsse.provider` with value `IBMJSSE2`.

Disabling a certificate

Disable a certificate that have already been imported into a keystore.

About this task

Certificates are enabled by default. You can disable a certificate as method for disabling partner access. You might also disable a certificate when preparing it as a replacement for a certificate that is about to expire. In this case, you don't want to enable the key until it replaces the existing key.

Procedure

1. Log on to the console.
2. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**. The Keystores panel opens.
3. Select a keystore or truststore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel opens.
5. Type your keystore password
6. Click **OK**.
7. Select a key from the keys table.
8. Click **Disable** to deactivate the key.

What to do next

If you are disabling the certificate so that you can prepare it as a replacement for another certificate that is in use, continue with the task for modifying your federation or partner properties in "Modifying certificate settings."

Modifying certificate settings

If you want to use a different certificate in your federation or in your partner's configuration, you must modify your federation or partner's key properties.

Before you begin

Before beginning this task, ensure that you have obtained new certificates and received or imported them into the appropriate keystore or truststore. See "Replacing an existing certificate" on page 97 for details.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on**.
3. Click either of the following:
   - **Federations**, if you want to specify a different signing or decryption certificate.
   - **Partners**, if you want to specify a different validation, encryption, server validation, or client certificate.

   The panel shows a list of configured federations or partners.
4. Select the federation or partner you want to update.
5. Click **Properties** to view the currently configured properties. The properties shown on this panel are described in the online help.
6. Locate the sections in which the certificates are defined and change the certificates named in those fields to the names of the certificates you want to use. For example, you must change the certificate that is specified in any of the following Properties sections:
   - **Signature Options** or **Signatures**
   - **Encryption**
   - **SOAP SSL Connection Parameters**, (only in the Partner Properties), which include:
     - **Server Validation Certificate**
     - **Client Certificate**
7. When you have finished modifying properties, click **OK** to close the Federation Properties panel.

What to do next

Enable the certificates as described in "Enabling a certificate."

Enabling a certificate

Certificates are enabled by default. You must enable a certificate only if you have previously disabled it.

About this task

You have to identify the keystore where you have imported the certificate to before you can enable the certificate.

To enable a certificate:

Procedure

1. Log on to the console and click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystore**. The Keystores panel opens.
2. Select a keystore or truststore from the Keystore table. The **View Keys** button is activated.
3. Click **View Keys**. You are prompted for a password.
4. Enter the keystore or truststore password. The Keys panel opens and the certificates are listed.
5. Select a certificate from the keys table.

6. Click **Enable** to enable the certificate.

What to do next

If you have enabled a certificate that replaces another certificate in your keystore or truststore, consider removing the certificate you no longer use, as described in "Deleting a certificate." If you have replaced a certificate that contains a public key that you must share with your partner, continue with "Exporting a certificate."

Deleting a certificate

Use this task to remove a certificate that is no longer needed. For example, if you have replaced a certificate in your federation or in your partner's configuration, you might want to delete the certificate that was replaced.

Before you begin

Attention: Before beginning this procedure, ensure that you are not deleting a certificate that is actively being used in a federation.

Procedure

1. Log on to the console
2. Click **IBM Tivoli Federated Identity Manager** > **Configure Key Service** > **Keystores**. The Keystores panel opens.
3. Select a keystore or truststore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. You are prompted for a password.
5. Enter the keystore or truststore password. The Keys panel opens and the certificates are listed.
6. Select a key from the keys table.
7. Click **Delete**. A message box prompts you to confirm that you want to delete the specified key.
8. Click **OK** to delete the key or click **Cancel** to exit from the window without deleting the key.

Exporting a certificate

You can export certificates from one keystore to put in another keystore, to back up your certificates, or to share the exported public key in the certificate with your partner. You can choose to export your public/private key pair in the certificate or only your public key.

About this task

If you are moving one of the certificates of your partner, work with the truststores where you store your partner certificates.

If you are exporting a certificate to share a public key with your partner or making a backup of your certificates, work with your keystore.

Attention: If you are exporting certificates to share them with your partner, be sure to export *only* your public keys.

Procedure

1. Log on to the console.
2. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**. The Keystores panel opens.
3. Select a keystore or truststore from the Keystore table. The **View Keys** button is activated.
4. Click **View Keys**. You are prompted for a password.
5. Enter the keystore or truststore password. The Keys panel opens and the certificates are listed.
6. Select a certificate from the keys table.
7. Click **Export**. The Export Key panel opens.
8. Select an export format.

   JKS
   Java Keystore

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

   PEM
   Privacy-Enhanced Message: public certificates.
9. (Optional) Select **Include Private Key**. Include your private key only if you are making a backup of your public/private key pair, or if you are moving your public/private key pair to a different keystore. *Do not* export your private key and share it with your partner.
10. Click **Download Key**.
11. When prompted, enter a file name for the new file. For example:

    `testkey.jks`

    Retain knowledge of the key location, so that you can import it elsewhere or share it with your partner.
12. Click **Cancel** to exit the Export Key portlet.

Using cryptographic hardware for a keystore

A cryptographic hardware device can be used purely as an accelerator. It can also be used as both an accelerator and as a keystore.

The device provides a secure facility for storing keys and certificates. IBM Tivoli Federated Identity Manager components use Key Encryption Signature Service (KESS) for runtime signature creation and validation. KESS centralizes key and certificate management in addition to encryption and signature services.

KESS supports the PKCS#11 interface to the cryptographic hardware through the com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl class that is packaged with WebSphere Application Server version 6.1 to provide hardware cryptography and keystore support.

If you configure IBM Tivoli Federated Identity Manager to use the hardware cryptography support, the following functions are provided:

- Signing XML documents
- Validating signatures on XML documents
- Encrypting and decrypting elements of an XML document.

See the link to the hardware and software requirements in the IBM Tivoli Federated Identity Manager information center at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html

It contains information about the cryptographic hardware devices that are supported.

Setting up to use the cryptographic hardware

You can set up your cryptographic hardware to be used as a keystore.

Procedure

1. Install and configure your cryptographic hardware device according to the instructions provided by the manufacturer.

   In a clustered environment, you must:

   a. Install the cryptographic device on every node in the cluster and use the same password on all nodes.
   b. Configure the same keys and certificates in each hardware keystore. The path to the hardware device configuration file must be the same for each node because etc/kessjks.xml is replicated to each node.
   c. Be sure all key labels are the same in all hardware keystores.
   d. After a single sign-on partner is configured at the deployment manager of the cluster, export the public keys of your partner from the hardware keystore of the deployment manager. Then import the keys, using the same key labels, into the hardware keystore at each node in the cluster.
2. Next, create a configuration file that contains settings for the device. See "Creating a configuration file."

Creating a configuration file

Use the template configuration files that IBM Tivoli Federated Identity Manager provides to create configuration files for hardware cryptography devices.

About this task

In this procedure is an example of modifying the template configuration file for nCipher.

Procedure

1. Use the default configuration file that is provided with IBM Tivoli Federated Identity Manager as a template:
   - For UNIX platforms, /opt/IBM/FIM/etc/nipher_gen2.cfg.jsse
   - For Windows platforms, c:\Program Files\IBM\FIM\etc\nipher_gen2.cfg.jsse

     if the IBM Tivoli Federated Identity Manager was installed in c:\Program Files directory
2. Update the configuration file for your environment. Be sure to change the **library** parameter to be the path to the hardware device driver:
   - For the UNIX platforms, /opt/nfast/toolkits/pkcs11/libcknfast.so.
   - For Windows platforms, c:\nfast\toolkits\pkcs11\cknfast.dll

     if the hardware drivers were installed in c:\nfast.

Note: The template configuration file is set up with the algorithms and parameters required by IBM Tivoli Federated Identity Manager and changes to these parameters could result in runtime failures.

Configuring hardware cryptographic device

You can configure your hardware cryptographic device in the Integrated Solutions Console.

Before you begin

Ensure that you have already set up your cryptographic hardware and have created a configuration file for the devices.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager > Configure Key Service > Hardware Cryptographic Device** The Hardware Cryptographic Device Settings portlet opens.
3. Select **Use hardware cryptographic device** to activate use of a device
4. Enter a path to the configuration file for the device.
5. (Option) Enter a description for the device.
6. Supply the password for accessing the device.
7. Click **Load configuration changes to Tivoli Federated Manager runtime**.

Chapter 9. Managing the SSL configuration

During the configuration of your IBM Tivoli Federated Identity Manager environment, you might have configured SSL with server authentication to ensure that messages are secure as they are communicated between you and your partners. In addition, you might have configured mutual authentication, by requiring a client certificate from your partner, or if you are a client, you might have configured a client certificate.

Location of certificates and keys

In your IBM Tivoli Federated Identity Manager environment, some keys and certificates are stored in WebSphere Application Server keystores and truststores and some are stored in the IBM Tivoli Federated Identity Manager keystores and truststores. The location depends on the purpose of the keys and certificates being used. For the purposes of SSL communication, the following keystores and truststores are used:

Certificates and keys	Location
SSL server certificates and their private keys	WebSphere Application Server *keystore* of the partner who acts as the server in the SSL communication.
CA certificate for clients that presents a client certificate to the server. (This is the public key of the client.)	WebSphere Application Server *truststore* of the partner who acts as the server in the SSL communication.
CA certificate for servers that have SSL server authentication configured. (This is the public key of the server.)	IBM Tivoli Federated Identity Manager *truststore* of the partner who acts as the client in the SSL communication.
SSL client certificates (those used for client certificate authentication) and their private keys.	IBM Tivoli Federated Identity Manager *keystore* of the partner who acts as the client in the SSL communication.

More information about configuring SSL in your environment is provided in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

The following tasks can help you manage your existing SSL configuration and the certificates and keys described in the preceding table:

Viewing your server SSL settings

Review the SSL settings on your server, as part of your IBM Tivoli Federated Identity Manager environment maintenance.

About this task

When you install WebSphere Application Server 6.1 and IBM Tivoli Federated Identity Manager, you have two default SSL configurations 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 IBM 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.

Procedure

1. Log on 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 review. For example, click **NodeDefaultSSLSettings**.
5. Ensure that the **Keystore name** shows the keystore where your certificate is stored.
6. Click the **Get certificate aliases** button to ensure that all certificate aliases in your keystore shows.
7. In the **Default server certificate alias** field, ensure that the correct certificate is specified.
8. Click the name of the other SSL configuration you want to review. For example, click **FIMSOAPEnpointSSLSettings**.
9. Ensure that the **Keystore name** shows the keystore where your certificate is stored.
10. Click the **Get certificate aliases** button to ensure that all certificate aliases in your keystore shows.
11. In the **Default server certificate alias** field, ensure that the correct certificate is specified.
12. Click **OK**.

Replacing the SSL server certificate

Server certificates usually have an expiration date. Replace your server certificate before it expires.

Replacing the certificate involves:

1. "Creating a certificate request" on page 115
2. "Receiving a signed certificate issued by a certificate authority" on page 116
3. "Associating a certificate with your SSL configuration" on page 117

After you have successfully replaced the certificate, extract the certificate and share it with your partner so that your partner can access the server. See “Extracting a certificate to share with your partner” on page 118 for details.

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 8.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/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 on 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**.
15. 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.

16. 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 required by the certificate authority to whom you will make your request.
17. 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 safekeeping.

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 115. The certificate must have also 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, see 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 on 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**
9. Click **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

Associate the certificate with your SSL configuration.

Associating a certificate with your SSL configuration

After adding a signed certificate to your keystore, you must associate your server SSL configuration settings with that certificate.

About this task

When you install WebSphere Application Server 6.1 and IBM Tivoli Federated 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 IBM 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 115 and "Receiving a signed certificate issued by a certificate authority" on page 116 to create and receive the additional signed certificate and repeat these instructions.

Procedure

1. Log on 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** shows the keystore where your certificate is stored.
6. Click the **Get certificate aliases** button to ensure that all certificate aliases in your keystore show.
7. In the **Default server certificate alias** field, select your signed certificate.

8. Click **Apply**
9. Click **Save** when prompted to save the configuration to the master configuration. The SSL configuration now uses the new certificate.

What to do next

Repeat these steps to associate the other SSL configuration with the appropriate certificate.

Sharing your server certificate with your partner

After you have added an SSL certificate to your server, such as a signed CA certificate, you must share a copy of public key for that certificate with your partner.

You have two options for sharing the public key of your server with your partner:

- "Extracting a certificate to share with your partner"
- "Instructing your partner to retrieve a certificate from the console" on page 119

Extracting a certificate to share with your partner

After you have added an SSL certificate to your server (such as a signed CA certificate), you must share a copy of public key for that certificate with your partner. One method is to extract the public key from your server certificate and then send 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
5. Click **Extract** in the upper-right corner.
6. 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`
7. Select the default data type from the list.
8. Click **Apply**.
9. Click **Save**. The signer portion of the personal certificate is stored in the .arm file that you specified.

What to do next

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 uses IBM Tivoli Federated Identity Manager, the partner must import your certificate into its IBM Tivoli Federated Identity Manager truststore.

Instructing your partner to retrieve a certificate from the console

After adding an SSL certificate to your server (such as a signed CA certificate), you must share a copy of the public key for that certificate with your partner. If your partner is using IBM Tivoli Federated Identity Manager, your partner can retrieve the public key through the IBM Tivoli Federated Identity Manager console.

Before you begin

Provide the following instructions *to your partner*.

Procedure

1. Ensure that you have prepared a truststore for storing the certificate. See the keystore topics in the *IBM Tivoli Federated Identity Manager Configuration Guide*.
2. Log on to the console.
3. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystore**.
   The Keystores panel opens.
4. Select the truststore where you want to store the certificate in the Keystore table. The **View Keys** button is activated.
5. Click **View Keys**. The Password panel opens.
6. Type your truststore password.
7. Click **OK**.
8. Click **Retrieve Certificate from SSL Connection**.
9. Complete the fields to specify the host name and port name from which you get the certificate. Optionally, click the **Show Signer Info** to view the certificate before retrieving.
10. Complete the **Alias** field with the name you want to use for the certificate.
11. Then, click **OK**. The certificate is added to the truststore.

Replacing your client certificates

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

If you are the client, use the following procedures to replace certificates, as needed:

- "Retrieving the server certificate from your partner"
- "Obtaining your client certificate" on page 120

Retrieving the server certificate from your partner

If your partner has server authentication configured, you need the public key from that server certificate and you store it in a truststore used by your IBM 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.

Procedure

1. Log on to the console.
2. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**.
   The keystores panel opens.
3. Select the truststore where you want to store the certificate in the keystore table. The **View Keys** button is activated.
4. Click **View Keys**. The Password panel opens.
5. Type your truststore password.
6. Click **OK**.
7. Click **Retrieve Certificate from SSL**.
8. Complete the fields to specify the host name and port name from which you can retrieve the certificate.
9. (Optional) Click the **Show Signer Info** to view the certificate before retrieving.
10. Complete the **Alias** field with the name you want to use for the certificate.
11. Click **OK**. The certificate is added to the truststore.

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 must 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.

Procedure

1. Request a public/private key pair certificate from a certificate authority (CA):
   a. Log on to the console.
   b. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**.
      The Keystores panel opens.
   c. Select a keystore from the Keystore table. The **View Keys** button is activated.
   d. Click **View Keys**. The Password panel opens.
   e. Type your keystore password.
   f. Click **OK**
   g. Click **Create Certificate Request**. The Create a certificate request panel opens.
   h. Complete the fields.
   i. Click **OK**. The Generated Certificate Signature Request window opens.
   j. 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.

k. 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 must be replaced with the signed certificate from the CA.

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

When you have created all of your certificate sign requests, follow your CA's instructions for transmitting the request file. 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 on to the console.
   b. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**.
      The Keystores panel opens.
   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 opens.
   e. Type your keystore password.
   f. Click **OK**.
   g. Click **Receive Certificate from CA**.
   h. Select the location of the certificate that you received from the CA.
   i. Click **OK**. The temporary self-signed certificate in the keystore is replaced with the received signed certificate.
3. Provide the public key for this certificate to your partner:
   a. Log on to the console.
   b. Click **IBM Tivoli Federated Identity Manager** > **Key Service > Keystores**.
      The Keystores panel opens.
   c. On the Keystores panel, select a keystore from the Keystore table. The **View Keys** button is activated.
   d. Click **View Keys**. The Password panel opens.
   e. Type your keystore password.
   f. Click **OK**.
   g. Select the keys you want to export and click the **Export** button. The Export Key panel opens.
   h. 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

   i. Ensure that the **Include Private Key** check box is *not* selected. Only you should have your private key.
   j. Click **Download Key**.
   k. When prompted, enter a file name for the exported key.
      For example: `mypublickey.pem`
      (Optional) Click **Browse** to find the file on the file system.
   l. Click **Cancel** to exit.

What to do next

Provide the certificate to your partner. The partner must have the following items:

- The truststore must have the CA certificate from the CA who issued your certificate.
- The server must have the capability to get the certificate revocation list of the CA.

Replacing the client certificate of your partner

In your initial configuration, you might have required that your partner provide a client certificate to authenticate to your server. If the client certificate of your partner has expired or changed, you must replace it on your WebSphere Application Server.

Before you begin

Before continuing with this task, ensure that you are replacing a certificate in an existing client certificate configuration. The tasks related to the initial configuration, include:

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

See the *IBM Tivoli Federated Identity Manager Configuration Guide* for information.

Before replacing the certificate, complete the following tasks:

- Ensure you have the public key certificate for the client certificate for your partner to 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 to permit access to the endpoint by authenticated users individually or authenticated users who are part of specific groups.

Procedure

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

   Note: In these instructions the certificate of the partner is named `partnerca.pem` and the directory to which the certificate was copied is named `/tmp`.
2. Log on to the console.
3. Click **Security** > **SSL Certificate and Key Management**.
4. Select **Key stores and certificates**.
5. Select the WebSphere Application Server truststore, such as **NodeDefaultTrustStore**.
6. Click **Signer certificates**.
7. Click **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. Ensure that you have completed these steps:
    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 to allow individual authenticated users or authenticated users who are part of specific groups.
       - If you require client certificate authentication from individual users, repeat the user name creation step 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 the user name creation step 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. Ensure that the SOAP authentication settings in the IBM Tivoli Federated Identity Manager console are correct:
    a. Log on to the console.
    b. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
    c. Select the point of contact server that you are using in your environment.
    d. Click the **Advanced** button. The SOAP Endpoint Security Settings panel opens.
    e. 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.
    f. Select **Client Certificate Authentication**.
    g. Click **OK**.
    h. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

Chapter 10. Modifying the alias service database settings

You can modify the database setting for the alias service.

Before you begin

Before modifying the database settings, be sure that you have set up your database as required. Refer to the topics about setting up the alias service database in the *IBM Tivoli Federated Identity Manager Configuration Guide* for more information.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Alias Service Settings**. The Alias Service Settings portlet opens.
3. Select the appropriate option for your configuration.
   - **JDBC Provider and Data Source**

     Use this option if you must use a JDBC database to store name identifier information.
   - **LDAP**

     Use this option if you must use an LDAP database to store name identifier information. If you choose this option, you must specify additional properties:

     a. Specify the properties for the alias service to use when searching the LDAP user registry.

     b. Specify communication properties for the alias service to use when communicating with LDAP servers.

     c. Specify configuration parameters for each LDAP server.

     See the online help for descriptions of the properties.
4. Click **Apply**.
5. Click **OK**.

Chapter 11. Managing audit settings

You can audit IBM Tivoli Federated Identity Manager events to monitor system activities.

For additional information about auditing, see the *IBM Tivoli Federated Identity Manager Auditing Guide*.

Use the Audit Profile Client table in the console to perform the following auditing management tasks:

Enabling or disabling auditing

Auditing is not enabled by default. If you want to monitor system events, you must enable auditing.

About this task

Enabling and disabling event auditing is separate from specifying auditing values, which is done in the administration console. You can set values without enabling auditing. After enabling auditing, you must specify the values.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing panel opens.
3. Select the **Enable audit** check box to enable auditing. To disable auditing, clear the check box.
4. Click **OK** when you are done.

What to do next

Be sure to review the audit file settings and modify them as appropriate. See "Modifying audit client profile" on page 128 for more information.

Activating audit client profiles

To enable an audit client profile, you must activate it.

About this task

Only one profile at a time can be active on the system.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing Settings panel opens.
3. When auditing is not enabled, select the **Enable audit** check box to enable it.
4. In the Audit Profile table, click the applicable profile in the **Select** column.
5. Click **Make Active**. The selected audit profile becomes activated.
6. Click **Apply** to store the changes for the configuration file. The software shows a confirmation message.
7. Click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes. The changes take effect immediately. A check mark in the Current Profile column of the Audit Clients Profiles table indicates that the selected profile is now active.

Deleting audit client profile

To remove an audit client profile from the Audit Client Profile table, you must delete it.

About this task

The Delete button becomes disabled if you select the current (active) profile.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing Settings panel opens.
3. If auditing is not enabled, select the **Enable audit** check box.
4. In the Audit Profile table, click the applicable profile in the **Select** column.
5. Click **Delete**.
6. Click **OK** to confirm. The software shows a confirmation message.
7. Click **Load confirmation changes to Tivoli Federated Identity Manager runtime** to reload your change, which takes effect immediately.

Modifying audit client profile

Review the audit file settings and modify them as required.

About this task

You can view and modify the properties of all audit client profiles in the Audit Client Profile table.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing Settings panel opens.
3. If auditing is not enabled, select the **Enable audit** check box to enable it.
4. In the Audit Profile table, click the applicable profile in the **Select** column.
5. Click **Properties**. The following panels open:

- Profile Name: Use to modify name and description, and to select an Audit Event Handler.
- Audit Clients: Use to modify the Audit Event Handler settings.

6. Click **Apply** to store the changes for the configuration file. The software shows a confirmation message.
7. Click **Load confirmation changes to Tivoli Federated Identity Manager runtime** to reload your changes.

Creating audit client profile

A wizard assists you in creating a customized Audit Client module for IBM Tivoli Federated Identity Manager audit events.

Before you begin

Before you begin, complete the following tasks:

- Publish the custom Audit Event Handler plug-ins to the runtime mode.
- Gather the parameter names and values that are to be passed to the plug-in.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing Settings panel opens.
3. If auditing is not enabled, select the **Enable audit** check box to enable it.
4. In the Audit Profile table, click **Create**. A wizard opens.
5. Enter the name and an optional description for the customized audit profile.
6. Select the appropriate event handler of the profile from the **Audit Event Handler** menu.

   The following pre-configured event handler options are available:

   - **CBEXMLAuditEventHandler**: Specifies a CBE 1.1 file-based event handler. The Common Base Event (CBE) is a standard format for event records. This implementation, unlike the CARS-based event handler, does not have a 2 K size limitation on audit records. Consequently, IBM Tivoli Federated Identity Manager can include additional information about some audit events.

     If you choose this event handler, specify these additional settings:

     a. Enter the directory location of the audit log file in the **Audit log location** field. The audit files are located in this directory.
     b. (Optional) Enter an appropriate name in the **Audit log file name** field.
     c. In the **Maximum audit file size before rollover (MB)** field, specify the maximum size of the audit file in megabytes.
     d. In the **Maximum number of audit files before rollover** field, specify the maximum number of audit files before overwriting.

   - **Tivoli Common Auditing and Reporting Service (CARS)**: Provides the text-file-based and WebService-based CARS Audit Event Handler. The CARS profile is the event handler implementation that uses the CARS solution for consuming the generated events. CARS provides a file-based audit records repository or a CARS repository accessible by WebServices request. With this profile, you can integrate IBM Tivoli Federated Identity Manager events to your CARS-based event handler solution for centralized event handling. This implementation has a 2 K size limitation on audit records.

     If you choose this event handler, select one of the following options:

– **Audit file (local file)**. If you select this option, specify the additional properties:
  a. In the **Audit log location** field, enter the directory location of the audit log file. The audit files are located in this directory.
  b. In the **Maximum audit file size before rollover (MB)** field, specify the maximum size of the audit file in megabytes.
  c. In the **Maximum number of audit files before rollover** field, specify the maximum number of audit files before overwriting.
– **Tivoli Common Auditing and Reporting Service**. If you select this option, specify the additional properties:
  a. In the **Web Service URL** field, enter the appropriate URL. You can use either a secure (HTTPS) or non-secure (HTTP) connection.
  b. In the **Disk cache location** field, specify the directory location for the disk cache files. The disk cache records audit events until they can be sent to CARS. The default directory location of the disk cache (on Linux) is: http://localhost:9080/CommonAuditService/services/EmitterDisk cache location.
  c. Click **Web Service Security Settings** to configure the Secure Socket Layer (SSL) and Authentication settings. See the online help for descriptions of the properties.

Note: You can also deploy a custom-coded event handler. To deploy a custom-coded event handler, implement the AuditEventHandler interface and deploy it as an OSGI extension.

7. Click **Finish** to store the settings for the new audit client profile. The software shows a confirmation message.
8. Click **Load confirmation changes to Tivoli Federated Identity Manager runtime** to reload your changes.

Modifying audit events

Several component events are selected to be audited by default. You can modify the setting for each component by clearing or selecting its corresponding check box.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Auditing**. The Auditing Settings panel opens.
3. Click **Audit Events**. A list of events with corresponding check boxes shows. A check mark indicates that the event will be audited. See the online help for descriptions of the events.
4. Clear or select the check boxes to specify the events you want to audit.
5. Then click **OK**.
6. Click **Audit Settings** to ensure that you have enabled auditing as described in "Modifying audit client profile" on page 128.

Chapter 12. Managing application server configuration

If your target application is hosted by a server that is separate from the WebSphere Application Server where IBM Tivoli Federated Identity Manager is installed, you must perform several configuration tasks on that server. For example, if your server requires the use of a Web plug-in component of IBM Tivoli Federated Identity Manager, you must configure the plug-in before it can be used in your environment.

Configuration tasks for Web servers include topics such as:

- Configuring the LTPA cookie
- Defining attributes for the LTPA token
- Disabling the automatic generation of LTPA keys
- Creating the plug-in configuration file
- Copying the plug-in configuration file to the application server

For details about these topics, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

For preliminary information about exporting and updating LTPA keys, modifying an existing configuration or modifying the existing plug-in log settings, see the following topics:

- "Exporting the LTPA key from the point of contact server"
- "Importing the LTPA key to the WebSphere Application Server" on page 132, if you are using a separate WebSphere Application Server to host your target application
- "Updating the LTPA key on the plug-in server" on page 133, if you are using a server that requires a plug-in
- "Modifying plug-in configuration manually" on page 133 or "Modifying the plug-in configuration file using the console" on page 139
- "Copying the plug-in configuration to the server" on page 140
- "Modifying the log settings of a plug-in" on page 140

Exporting the LTPA key from the point of contact server

If you use 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 IBM Tivoli Federated Identity Manager plug-in is installed, you must 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 must import the key might mistakenly interpret that key to be expired.

Procedure

1. Log on 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 it with the server that is hosting your application. See the appropriate topic for the server you are using:

- "Importing the LTPA key to the WebSphere Application Server," if you are using a separate WebSphere Application Server to host your target application.
- "Updating the LTPA key on the plug-in server" on page 133, if you are using a server that requires a plug-in.

Importing the LTPA key to the WebSphere Application Server

If your target application is hosted by 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 the separate 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 key from the location where they were exported to a location on your target application server.
- Obtain the password for the LTPA key. A password was assigned to the key when it was exported from the WebSphere point of contact server.

Procedure

1. Log on to the console on the *target server*. Do not log on to your IBM 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 key. This password must match the password that was used when the key was exported.
4. In the **Fully qualified key file name** field, specify the fully qualified path to the location where the LTPA key is located. You must have write permission to this file.
5. Click **Import keys** to import the key.
6. Click **OK**.

7. Click **Save** to save the changes to the master configuration.

Updating the LTPA key on the plug-in server

If your target application is hosted by a server that requires a plug-in, such as an IHS server or IIS, that is separate from your WebSphere point of contact server, you must copy the LTPA key from the point of contact server onto that server.

Before you begin

Before you continue with these steps, ensure that you have completed the following tasks:

- Exported the LTPA key from your point of contact server, as described in "Exporting the LTPA key from the point of contact server" on page 131.
- Verified that the time and date on the point of contact server and your server where the plug-in is installed 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 IBM HTTP or Apache Server:

   ```
   /opt/IBM/FIM/webpi/etc
   ```

   On an IIS server:

   ```
   C:\Program Files\IBM\FIM\webpi\etc
   ```

Modifying plug-in configuration manually

If you have previously configured a plug-in using the tasks in the *IBM Tivoli Federated Identity Manager Configuration Guide*, you can modify that configuration by either reconfiguring the file using the IBM Tivoli Federated Identity Manager console or by manually editing the file on the server where the plug-in is installed. The following steps describe how to modify the configuration file manually.

Before you begin

Note: If you must update the password for the LTPA key file, you must use the console to modify the configuration.

Before beginning this task, ensure that you have completed all of the LTPA and plug-in configuration tasks in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

About this task

The plug-in configuration file is an XML file that is created using the IBM Tivoli Federated Identity Manager console. By default, the file is named `itfimwebpi.xml` and must be copied from the server on which it was created to the server where the plug-in is installed and pasted in the plug-in installation directory. For example:

On an IBM HTTP or Apache Server:

```
/opt/IBM/FIM/webpi/etc
```

On an IIS server:

```
C:\Program Files\IBM\FIM\webpi\etc
```

The file schema and its description are provided in "Web plug-in configuration file schema."

Use an XML editor or a text editor to modify the file. Consider making a backup copy of the file before you begin the following task.

Procedure

1. On the server where the plug-in is installed and the configuration file has been copied, use an XML or text editor to open the file.

   On an IBM HTTP or Apache Server:

   ```
   /opt/IBM/FIM/webpi/etc/itfimwebpi.xml
   ```

   On an IIS server:

   ```
   C:\Program Files\IBM\FIM\webpi\etc\itfimwebpi.xml
   ```

2. Modify the configuration settings that are specified in the file as appropriate to your environment.
3. Save and close the file.

What to do next

Restart the web server where the plug-in is installed for the changes to take effect.

Web plug-in configuration file schema

The itfimwebpi.xml file is an XML file that contains configuration information that is specific to your Web server and your IBM Tivoli Federated Identity Manager environment.

Configuration file example

```
<?xml version="1.0" encoding="UTF-8"?>
<pi:WebPIConfiguration
    enabled="true"
    serial="1184026004328"
 version="6.2.0"
 cacheSize="32768"
 xmlns:fim="urn:ibm:names:ITFIM:1.0"
 xmlns:pi="urn:ibm:names:ITFIM:1.0:webpi"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ibm:names:ITFIM:1.0:webpi webpi.xsd urn:ibm:names:ITFIM:1.0 itfim.xsd ">
 <pi:LoggingConfiguration
      id="uuid-6cd36762-3404-11dc-8314-0800200c9a66"
      level="INFO"
      logFilename="/itfimwebpi.log" />
 <pi:LTPAConfiguration id="uuid-6cd36543-3404-42fg-8314-0800200c9a66"
                      ltpaPassword="{obf0}iogiodfjksdfskjfsf==">
 </pi:LTPAConfiguration>
 <pi:Applications>
  <pi:Application id="uuid-6cd35678-3404-34gf-8314-0800200c9a66"
      name="App1Name" url="http://tempuri.org">
   <pi:Description>This is App 1's description</pi:Description>
   <pi:CookiesToStrip>
    <pi:CookieToStrip cookieName="JSESSIONID" id="uuid-6cd38876-3404-
      98uy-8314-0800200c9a66" />
    <pi:CookieToStrip cookieName="PD_SESSION_ID" id="uuid-6cd97854-3404-87uy-8314-0800200c9a66" />
   </pi:CookiesToStrip>
   <pi:Headers>
    <pi:Header
      ltpaAttributeName="u"
      headerName="iv-user"
      id="uuid-6cd6453-3404-54tg-8314-0800200c9a66"
      stripClientHeader="true" />
    <pi:Header
      ltpaAttributeName=""
      headerName="iv-cred"
      id="uuid-6cd76543-3404-76hg-8314-0800200c9a66"
      stripClientHeader="true" />
```

```
   </pi:Headers>
   <pi:ServerVariables>
    <pi:ServerVariable ltpaAttributeName="REMOTE_USER" id="uuid-6cd6543-3404-76fd-8314-0800200c9a66"
     variableName="u" />
   </pi:ServerVariables>
  </pi:Application>
 </pi:Applications>
</pi:WebPIConfiguration>
```

Attributes and configuration elements

The following attributes are used in a completed configuration file. The attributes are grouped by the configuration element in the file.

WebPIConfiguration

enabled=*true*| *false*
Enables or disables the functionality of the plug-in. The default value is false.

cacheSize=*number_of_entries*
The maximum number of entries in the LTPA token cache. If zero is specified, then no caching occurs. The default value is 0. The maximum size is 32767.

LoggingConfiguration

level=*ERROR*| *WARN*| *INFO*| *DEBUG*| *NONE*
The logging level:

- ERROR
- WARN
- INFO
- DEBUG
- NONE

The default value is INFO.

logFileName=*path_and_file_name*
The log file name. The path defined here must be an absolute path. The default path is:

On an IBM HTTP or Apache Server:
`/opt/IBM/FIM/webpi/etc/itfimwebpi.log`

On an IIS server:
`C:\Program Files\IBM\FIM\webpi\etc\itfimwebpi.log`

Note: The Web server must have write access to this file. If it does not have write access and logging is enabled, no log entries show in the log file.

LTPAConfiguration

ltpaPassword=*password*
The obfuscated password used to decode the LTPA keys file.

Note: The console creates an obfuscated form of the password.

Applications

Application
Entries with application-specific configuration. This element can be left blank if no specific configuration is required. The following elements can be used with the Application element:

Name=*name*
The name that identifies the application. This name is used in the console and the log files.

url=*URL*
The URL path of the application URLs. The URL path should include the path and any query parameters. It does not include the protocol, host name, or port paths of the URL.

For example, a URL of http://www.example.com/myapp/myportal.html has a URL path of /myapp/myportal.html. The plug-in processes the most precise match.

The Web plug-in configuration contains zero or more URL regular expression patterns identifying applications. An application is said to match if its URL path matches one of the configured URL patterns. The Web plug-in adds LTPA token attributes as HTTP headers to web requests destined for matching applications.

The URL path /myapp/myportal.html would match the following URLs patterns:

- .*
  Matches all URL paths
- /myapp/.*
  Matches all URL paths for the myapp application
- /myapp/myportal.html
- Matches the specific application page

When a web request URL path matches more than one configured URL regular expression pattern, the most precise match (the longest regular expression) is used. For example, although all of the above URL patterns match the web request URL path, the last URL pattern would be considered the most precise match.

Description
The application description.

Headers
Provides a description of how LTPA attributes are used to create HTTP headers. The following attributes can be used with the Headers element:

stripClientHeader=*true*| *false*
True removes any existing headers with the name specified by **headerName** before it looks for LTPA attributes. False does not remove any headers with the name specified by **headerName**.

Attention: When this setting is set to true, headers are removed regardless of the presence of an LTPA cookie. Removal of the header guarantees that the application only receives the headerName when created from ltpaAttributeName. Use of stripClientHeader=true prevents an untrusted client from sending a header that should ONLY be derived from the LTPA token. The application can then assume that the header value is trustworthy.

headerName=*headername*
The name of the HTTP header that is created with the value of the LTPA attribute.

ltpaAttributeName=*attributename*
The LTPA attribute whose value is used to create an HTTP header. If the LTPA attribute does not exist, the HTTP header is not created or modified.

CookiesToStrip
HTTP cookies to remove from the Web request if the request does not contain a valid LTPA token cookie.

cookieName=*name*
The name of the cookie to remove from the request if the request does not contain a valid LTPA token cookie.

ServerVariables
Server variables to set.

Note: Server variables are not created on IIS.

variableName=*name*
The name of the server variable.

ltpaAttributeName=*attributename*
The LTPA attribute whose value is used to create an HTTP header. If the LTPA attribute does not exist, the HTTP header is not created or modified.

Schema

The configuration schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:ibm:names:ITFIM:1.0:webpi"
    xmlns:pi="urn:ibm:names:ITFIM:1.0:webpi"
    xmlns:fim="urn:ibm:names:ITFIM:1.0"
    attributeFormDefault="unqualified" elementFormDefault="qualified">

    <xsd:import namespace="urn:ibm:names:ITFIM:1.0" schemaLocation="itfim.xsd" />

    <xsd:element name="WebPIConfiguration" type="pi:WebPIConfigurationType"/>

    <xsd:complexType name="WebPIConfigurationType">
        <xsd:complexContent>
            <xsd:extension base="fim:ConfigBaseType">
                <xsd:sequence>
                    <xsd:element name="LoggingConfiguration" type="pi:LoggingConfigurationType" minOccurs="1" maxOccurs="1" />
                    <xsd:element name="LTPAConfiguration" type="pi:LTPAConfigurationType" minOccurs="1" maxOccurs="1" />
                    <xsd:element name="Applications" type="pi:ApplicationsType" minOccurs="1" maxOccurs="1" />
                </xsd:sequence>
                <xsd:attribute name="enabled" type="xsd:boolean" use="required" />
                <xsd:attribute name="cacheSize" type="xsd:long" use="required" />
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:simpleType name="LoggingLevelType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="INFO"/>
            <xsd:enumeration value="WARN"/>
            <xsd:enumeration value="ERROR"/>
            <xsd:enumeration value="DEBUG"/>
        </xsd:restriction>
    </xsd:simpleType>

 <xsd:complexType name="LoggingConfigurationType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
                <xsd:attribute name="level" type="pi:LoggingLevelType" default="INFO" />
                <xsd:attribute name="logFilename" type="xsd:string" default="fimwebpi.log" />
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="LTPAConfigurationType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
             <xsd:attribute name="ltpaPassword" type="xsd:string" use="required" />
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ApplicationsType">
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" type="pi:ApplicationType" name="Application" />
        </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="ApplicationType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
                <xsd:sequence>
                    <xsd:element name="Description" type="xsd:string" minOccurs="0" maxOccurs="1" />
                    <xsd:element maxOccurs="unbounded" minOccurs="1" type="pi:CookiesToStripType" name="CookiesToStrip" />
                    <xsd:element maxOccurs="unbounded" minOccurs="1" type="pi:HeadersType" name="Headers" />
                    <xsd:element maxOccurs="unbounded" minOccurs="1" type="pi:ServerVariablesType" name="ServerVariables" />
                </xsd:sequence>
                <xsd:attribute name="name" type="xsd:string" use="required" />
                <xsd:attribute name="url" type="xsd:anyURI" use="required" />
                <xsd:attribute name="errorUrl" type="xsd:anyURI" use="optional" />
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="CookiesToStripType">
       <xsd:sequence>
           <xsd:element maxOccurs="unbounded" minOccurs="0" type="pi:CookieToStripType" name="CookieToStrip" />
       </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="HeadersType">
       <xsd:sequence>
           <xsd:element maxOccurs="unbounded" minOccurs="0" type="pi:HeaderType" name="Header" />
       </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="HeaderType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
          <xsd:attribute name="headerName" type="xsd:string" use="required" />
          <xsd:attribute name="ltpaAttributeName" type="xsd:string" use="required" />
             <xsd:attribute name="stripClientHeader" type="xsd:boolean" use="required"/>
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>
 <xsd:complexType name="CookieToStripType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
          <xsd:attribute name="cookieName" type="xsd:string" use="required" />
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>

 <xsd:complexType name="ServerVariablesType">
        <xsd:sequence>
            <xsd:element maxOccurs="unbounded" minOccurs="0" type="pi:ServerVariableType" name="ServerVariable" />
        </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="ServerVariableType">
        <xsd:complexContent>
            <xsd:extension base="fim:IdType">
          <xsd:attribute name="variableName" type="xsd:string" use="required" />
          <xsd:attribute name="ltpaAttributeName" type="xsd:string" use="required" />
            </xsd:extension>
        </xsd:complexContent>
 </xsd:complexType>
```

Modifying the plug-in configuration file using the console

If you have previously configured a plug-in using the tasks in the *IBM Tivoli Federated Identity Manager Configuration Guide*, you can modify that configuration by either reconfiguring the file using the IBM Tivoli Federated Identity Manager console or by editing the file on the Web server where the plug-in is installed. The following steps describe how to modify the configuration file using the console.

Before you begin

To complete this task, you 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 match to which HTTP header. For example, you might want to match the iv-cred HTTP header to the tagvalue_email LTPA attribute name.
- 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.

About this task

Modifying a configuration through the console is similar to creating a new configuration. If your existing configuration has many settings, such as several applications defined, consider modifying your configuration file manually. See "Modifying plug-in configuration manually" on page 133.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Web Server Plug-in Configuration**. The Web Plugin Single Sign-on Configuration panel opens.
3. Complete the information that is required for the server where the plug-in is installed in the **Web Server Plug-in Single Sign-on Configuration** and the **Web Server Plug-in Logging Configuration** sections. See the online help for descriptions of the fields.
4. Click **Save**.
5. In the **Web Server Plug-in Applications Configuration**, define an application to the single sign-on configuration by clicking the **Create** button. The Application Properties panel opens.
   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 application have been added.
- If you have completed the addition of the application to the server, click **OK**.

6. Click **Save**.
7. 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

Copy the updated configuration file to the Web server, as described in "Copying the plug-in configuration to the server."

Copying the plug-in configuration to the server

After you have modified the plug-in configuration file, you must copy that configuration to the Web server where the plug-in is installed.

Procedure

1. Locate the configuration file that you created using the steps in "Modifying the plug-in configuration file using the console" on page 139. The file is named `itfimwebpi.xml`.
2. Copy the file.
3. and then paste the file in the webpi directory on your Web server where the plug-in is installed:

   On an IBM HTTP or Apache Server:
   `/opt/IBM/FIM/webpi/etc`

   On an IIS server:
   `C:\Program Files\IBM\FIM\webpi\etc`

What to do next

Restart your the server where the plug-in is installed for the changes to take effect.

Modifying the log settings of a plug-in

Events that occur when you use a IBM Tivoli Federated Identity Manager Web plug-in are logged in a log file on the server where the plug-in is installed.

About this task

You can modify the location of the log file and the type of log that should be created.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Web Server Plug-in Configuration**. The Web Plugin Single Sign-on Configuration panel opens.
3. Click the **Web Plugin Logging Configuration** tab. The current log settings show.
4. Modify the settings as appropriate for your environment. The options for the type of log to create are:

   INFO (the default setting)

   WARN

   ERROR

   DEBUG
5. Click **OK**.

Chapter 13. Managing domains

Domain configuration is typically set during the initial installation and configuration of IBM Tivoli Federated Identity Manager.

The configuration of a domain, as part of the overall deployment of IBM Tivoli Federated Identity Manager is described in the *IBM Tivoli Federated Identity Manager Configuration Guide.*

Connecting to an existing domain

You can connect the management console to the management service for an existing domain.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager → Domains**. The Domains portlet opens.
3. Click **Create**. The Domain wizard opens the Welcome panel.
4. Click **Next**. The Management Service Endpoint panel opens.
5. Enter values for the specified properties.
6. When finished, click **Next**. The WebSphere Security panel opens.
7. Specify whether WebSphere global security is enabled.
   - 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**.

   The WebSphere Target Mapping panel opens.
8. Select or enter the name of your server or cluster.
   - When the WebSphere environment consists of a single server, the panel opens a Server name menu, with a name such as server1. If the server name does not show in the menu, enter it in the field.
   - When the WebSphere environment consists of a WebSphere cluster, the panel opens the Cluster Name menu. This menu lists the names of clusters defined in the cell. Select the name of the cluster to use.
9. When finished, click **Next**. The Select Domain panel opens. The wizard detects that a domain exists and prompts you to connect to the domain.
10. Click **Next** to establish a connection to the domain.

    The Summary panel opens.
11. Review the information about the Summary panel.
12. Click **Finish**.

Results

The management console is now connected to the domain. You can now perform management tasks on the current domain.

Modifying the domain properties

You can modify the domain properties that were specified during the configuration of IBM Tivoli Federated Identity Manager.

Procedure

1. Log on to the console and click **Tivoli Federated Identity Manager** > **Domain Management** > **Domain Properties**. The Domain properties portlet is displayed.
2. Change the properties as appropriate for your domain. See the online help for descriptions.
3. Click **OK** when you are done.

Viewing domain information

A read-only view of domain information provides details about the IBM Tivoli Federated Identity Manager and WebSphere Application Server installations.

Procedure

1. Log on to the console.
2. Select **Tivoli Federated Identity Manager** > **Domains**. The Domain portlet opens.
3. Select the domain.
4. Click **Properties**. The Domain Properties portlet opens.
5. Click the **Domain information** tab. The details of the components installed in the domain are listed. See the online help for descriptions of the information shown.

   Note: If you have recently changed domain configuration and have not restarted the management service, and you are ready to restart the management service, you can click **Refresh Management Service**.
6. Click **OK** when you are done.

Activating a domain

Use the Integrated Solutions console to activate a domain. When your deployment includes more than one domain, and you want to manage a different domain, you can switch to that domain by activating it.

About this task

You can manage only one domain at a time from the IBM Tivoli Federated Identity Manager management console. The domain that you can manage is the *active* domain.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager → Domains**. The Domains portlet opens.
3. Select the radio button in the **Select** column for the domain.
4. Click **Make Active**. A message box is shows, indicating that all open Management pages must be closed before changing the current domain.

5. Close any open management pages.
6. Click **OK**.

   This message shows because when you switch domains, you might still have console pages open for the domain that is currently active. These pages must be closed before you change domains because you can no longer work with them when you have activated another domain. When there are no further changes to any open pages, the console closes the pages for you.

Results

The Manage Domains panel is redrawn. A check mark shows in the **Current Domain** column of the domain that you activated. The domain that was previously active is now deactivated.

Changing the current domain

Use the Integrated Solutions Console to change the current domain to manage another domain.

About this task

Some deployments have only one IBM Tivoli Federated Identity Manager domain to manage. Other deployments, such as those implemented for a Data Center model, use multiple domains. In the Data Center model:

- You can have multiple IBM Tivoli Federated Identity Manager installation instances. The IBM Tivoli Federated Identity Manager console supports centralized management of these installation instances through the ability to change the current domain.
- You can use the WebSphere Application Server Network Deployment manager to complete its own management tasks against more than one IBM Tivoli Federated Identity Manager installation.

The IBM Tivoli Federated Identity Manager management console shows the Current Domain portlet in the right panel during all administrative tasks. Use this portlet to change domains.

When you change the current domain, you are changing the configuration to make another domain active. This task accomplishes the same thing as activating a domain. See “Activating a domain” on page 144.

Procedure

1. Locate the Current Domain portlet at the top of the right panel.
2. Click **Change Domain**. The Domains panel opens.
3. Select the domain that you want to manage.

   You are prompted to make sure that all management pages for the current domain are closed. A message box shows.
4. Resolve the status of any management pages that are currently open:
   - If you have management pages open, and must complete other tasks first, click **Cancel**. Go to the open management pages, complete the tasks, and then return to this task.
   - If you do not have management pages open, or if you have management pages open but do not have any unfinished tasks, click **OK** at the prompt to close the page.

Deleting a domain

You must unconfigure all runtime nodes and undeploy the runtime before you delete a domain.

Before you begin

Before you remove a domain, complete the instructions in "Removing a runtime application from WebSphere Application Server" on page 161.

About this task

You can remove the domain from the *console only*, or from the *console and server*:

- When you remove the domain from the console only, the console deletes all the local configuration information that is necessary to connect to the domain from the computer that hosts the management console (the Integrated Solutions Console). The IBM Tivoli Federated Identity Manager configuration information that is located on the WebSphere Application Server or WebSphere cluster is *not* removed.
- When you remove the domain from both the console and the server, you delete both the local configuration information and the associated configuration information from the WebSphere Application Server or WebSphere cluster on which IBM Tivoli Federated Identity Manager is deployed.

Note: When upgrading to a new version of IBM Tivoli Federated Identity Manager, delete a domain from the console only.

Procedure

1. Click **Tivoli Federated Identity Manager → Domains**.

   The Domains portlet opens.
2. Select the radio button for the domain in the **Select** column.
3. Click **Delete**.

   A message box prompts you to choose between one of two deletion modes.
4. Select a deletion mode:
   - Delete from server and console.

     This action deletes all domain configuration from the properties files used by the IBM Tivoli Federated Identity Manager management console, and also deletes all configuration from the properties files used by WebSphere Application Server.
   - Delete from console.

     This action deletes all domain configuration from the properties files used by the IBM Tivoli Federated Identity Manager management console, but does not delete the domain configuration from the properties files used by WebSphere Application Server.

Results

When you select a deletion mode, the configuration information is removed. The Domains panel is redrawn and the entry for the specified domain no longer exists.

Chapter 14. Managing event pages

Event pages generate files that are displayed in response to events that occur during single sign-on requests. You can customize the appearance or content of the event pages or modify the geographic or language locale that should be used to display the files that are generated by the pages.

The customization of event pages is an advanced task that requires knowledge of how event pages generate and display information that is related to an event. See the topics about customizing single sign-on event pages in the *IBM Tivoli Federated Identity Manager Configuration Guide* for details.

Tasks for managing event pages include:

- "Modifying event pages"
- "Managing page locales"

Modifying event pages

Event pages generate files that are displayed in response to events that occur during single sign-on requests. You can customize the appearance or content of the event pages.

About this task

The customization of event pages is an advanced task that requires knowledge of how event pages generate and display information that is related to an event. See the topics about customizing single sign-on event pages in the *IBM Tivoli Federated Identity Manager Configuration Guide* for details.

To locate the event page management panel in the console:

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Event Pages**.
3. Complete the fields as appropriate for your customization. See the topics about customizing single sign-on event pages in the *IBM Tivoli Federated Identity Manager Configuration Guide* and the online help for details.
4. Click **Apply** or **OK**.
5. When you are done and want to make the changes immediately, click **Publish**. Otherwise, click **Close** and publish the pages later using the task described in "Publishing pages" on page 157.

Managing page locales

You can modify the geographic or language locale should be used to display the files that are generated by events that occur during a single sign-on request.

About this task

The modification of page locales are related to the customization of event pages. Both tasks are advanced and require knowledge of how event pages generate and show information that is related to an event. See the topics about customizing single sign-on event pages in the *IBM Tivoli Federated Identity Manager Configuration Guide* for details.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Event Pages**.
3. Click the **Page Locales** tab.
4. Complete the fields as appropriate for your customization. See the topic about creating a page locale in the *IBM Tivoli Federated Identity Manager Configuration Guide* and the online help for details.
5. Click **Apply** or **OK**.
6. When you are done and want to make the changes immediately, click **Publish**. Otherwise, click **Close** and publish the pages later using the task described in "Publishing pages" on page 157.

Chapter 15. Exporting and importing server configuration

The configuration export and import functions enable you to export the configuration of an existing IBM Tivoli Federated Identity Manager server and then import the data onto another server.

These functions can be helpful if you want to create a backup of your configuration or to create a high-availability environment to meet load balancing requirements. For information about configuring a high-availability environment, see *IBM Tivoli Federated Identity Manager Configuration Guide*.

Tasks include:

- "Exporting configuration"
- "Importing configuration" on page 150
- "Backing up and restoring a domain" on page 151

Exporting configuration

Use this task to export the IBM Tivoli Federated Identity Manager configuration. For example, use this task to make a backup of configuration properties, or use it to migrate or copy your IBM Tivoli Federated Identity Manager configuration to another system.

Before you begin

Note: When you export your configuration, only your domain and federation settings (such as keys and partner settings) are exported. This function does not export your WebSphere Application Server configuration.

You can also use the export configuration task when you are creating a high-availability environment.

For information about configuring a high-availability environment, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

You must first install and configure IBM Tivoli Federated Identity Manager before you can export a configuration.

About this task

When you export a IBM Tivoli Federated Identity Manager configuration, the complete configuration is automatically gathered into a JAR file. The file serves as a IBM Tivoli Federated Identity Manager archive file. For example:

```
fimconfig-20061011-114614-0500.jar
```

You can save this file anywhere that is accessible to the standard Save or Download window presented by your browser.

Note: This means that the archive file is saved on the computer that runs your browser, or on a networked file system that is accessible to the browser. When you want to access this archive file later, as part of the importing a domain configuration task, you must access the location where you saved the archive file.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Import and Export Configuration**. The Import and Export panels opens.
3. Click **Export Configuration**.
4. When prompted, specify the location where the configuration JAR file is to be saved.

   Note: The location you specify, so that you can access this file later when you must import the configuration.
5. Click **OK**.

Importing configuration

Use this task to import the IBM Tivoli Federated Identity Manager configuration. This task is useful when you want to restore a configuration or create a duplicate of the configuration as part of implementing a data center model.

Before you begin

You must first install IBM Tivoli Federated Identity Manager on the server where you import the exported configuration. Then, configure it to match the installation on your existing server.

If you have not yet created a IBM Tivoli Federated Identity Manager archive file for the domain that you want to import, you must do that first. See "Exporting configuration" on page 149.

The archive file must be placed in a location that can be accessed by the browser from which you are using the console. If you exported the configuration (created the archive file) from the same browser, the archive file should be accessible.

If, however, you created the archive file while using a browser on a different computer, you must move the archive file to a different location in order for your current browser to be able to import the configuration.

Procedure

1. Log on to the console
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Import and Export Configuration**. The Import and Export panels opens.
3. Select the **Domain Name** of the domain into which the configuration archive is to be imported.
4. In the **Configuration Archive** field, enter the fully qualified path name to the IBM Tivoli Federated Identity Manager configuration archive.

   For example:

   ```
   /tmp/fimconfig-20061011-114614-0500.jar
   ```

   This archive was previously created using the Export procedure on another server.

   (Optional) Use the **Browse** button to locate the file.
5. Click **Import Configuration**. The configuration is automatically imported.

Backing up and restoring a domain

Use the IBM Tivoli Federated Identity Manager to export the configurations of a domain for backing up purposes and import the configurations for restoring a domain.

About this task

This task exports all of the configuration information for a domain. See "Exporting configuration" on page 149 to export one domain configuration at a time.

This task imports all the files necessary to migrate an existing domain configuration (new hardware) into a new environment (when moving from a test deployment to a production deployment). See "Importing configuration" on page 150. The Import task does not import the entire configuration of a domain.

To import the entire configuration of the domain, such as when restoring an existing configuration into the same environment, you must perform additional steps. These additional steps enable you to perform disaster recovery when your hardware and network environment is unchanged.

Procedure

1. Install IBM Tivoli Federated Identity Manager into the environment to be restored.
2. Stop the WebSphere Application Server.
3. Create a directory with a name that matches the name of the domain you want to create. Place the directory in:

   `$WAS_HOME/profiles/`*`your_profile_name`*`/config/itfim`
4. Unpack the JAR file containing the exported configuration. Place it in the directory

   `$WAS_HOME/profiles/`*`your_profile_name`*`/config/itfim/`*`your_domain_name`*
5. Restart the WebSphere Application Server.
6. Using the IBM Tivoli Federated Identity Manager administration console, create a domain on the server with the same name as the directory created previously as *your_domain_name*.

   Note: No existing information is overwritten.

Chapter 16. Managing point of contact servers

When you configure your environment, you can use WebSphere Application Server as your point of contact server, or you can choose to develop a custom point of contact server. You can manage your choice of point of contact servers using the Point of Contact panel in the console.

Example: You can view the properties of a point of contact server profile, modify its settings, or, if you have added one or more custom point of contact servers to your environment, you can select a different point of contact server to be the active one in your environment. In addition, you can use the panel to add a custom point of contact server profile to your environment or delete an existing custom profile.

Attention: Creating a custom point of contact server requires that you complete several tasks before adding a new profile to the environment using the console. For information about creating a custom point of contact server and its profile, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

For preliminary information about modifying, viewing, deleting or activating a point of contact server, see the following topics:

- "Viewing the properties of a point of contact server"
- "Deleting a custom point of contact server" on page 154
- "Activating a point of contact server" on page 154
- "Modifying the WebSphere point of contact server settings" on page 154

Viewing the properties of a point of contact server

You can view the properties of all of the point of contact server profiles in your environment. In addition, if you have custom point of contact server profiles configured and you are logged in as the administrator who created those custom profiles, you can also modify the profile properties.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the point of contract server profile that you are using in your environment.
4. Click **Properties**. The Profile Name panel of the Point of Contact Properties window opens.
5. Click the other selections on the left to view the properties for the following events:
   - Sign In
   - Sign Out
   - Local ID
   - Authentication
6. Click **Cancel** when you are done.

Deleting a custom point of contact server

If you have configured a custom point of contact server but will no longer use it, you can delete it.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the point of contact server you want to delete.

   Note: You can delete only a custom point of contact server that you have created. The point of contact servers that are provided with IBM Tivoli Federated Identity Manager cannot be removed.
4. Click **Delete**. The point of contact server you selected is removed.
5. Click **OK**.

Activating a point of contact server

To enable a point of contact server as the active server in your environment, you must activate it.

About this task

To activate a custom point of contact server:

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the point of contact server you want to activate.
4. Click **Make Active**. The point of contact server you selected is activated and is used as the point of contact server in your environment.

Modifying the WebSphere point of contact server settings

When you install Tivoli Federated Identity Manager, you can choose to activate the WebSphere point of contact server profile. After initial configuration, you can modify the server profile settings.

By default, the settings of this profile are:

SOAP Endpoint Security Settings

SOAP Port: 9444

SOAP endpoint authentication type: Allow unauthenticated users access to SOAP endpoints.

SPNEGO authentication

Default setting: Disabled.

Modifying SOAP port and endpoint authentication settings

By default, the SOAP port is 9444 and the endpoint security settings are set to **Allow unauthenticated users access to SOAP endpoints**. Change the setting if you want to use a different port or option.

About this task

The "users" referred to on this panel and in this procedure are "service users"; that is, they are user IDs that are configured in the local user registry and are used to authenticate the servers that your service provider uses to retrieve a request with an artifact.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the point of contact server profile that you are using in your environment.
4. Click **Advanced**. The SOAP Endpoint Security Settings panel opens.
5. Ensure that the SOAP Port is correct in your configuration. If not, change the port number.
6. In the SOAP Endpoint Security Settings section, select the appropriate option for your configuration:
   - **Allow unauthenticated users access to SOAP endpoints**

     When you choose this option, no authentication is required for users to access the artifact resolution endpoint.
   - **Allow authenticated users access to SOAP endpoints**

     When you choose this option, authentication is required for users to access the artifact resolution endpoint.

     If you want to require service providers to authenticate before they have access to the artifact resolution port, you must create accounts in your WebSphere Application Server user registry for the service providers to use.
   - **Allow users in the specified group access to SOAP endpoints**

     If you choose this option, you must also specify the group name in the **Group Name** field.

     When you choose this option, you are configuring access to the artifact resolution endpoint to be limited to members of the specified group. Members of the group must authenticate to be granted access.

     When you want to require service providers to authenticate to gain access to the artifact resolution endpoint, you must ensure that each service provider has a user identity (account) configured in the user registry. Add each service provider to the specified group.
7. If you choose an option that requires authentication, you are prompted to select the authentication type.
   - Basic Authentication

     Authentication that requires your service provider partner to provide a user name and password.
   - Client Certificate Authentication

     Authentication that requires your service provider partner to present a certificate to establish a secure authenticated session.

8. Click **OK**.
9. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

Modifying SPNEGO authentication settings

If you are an identity provider partner that uses WebSphere as a point of contact server, and you use SPNEGO authentication, you must enable it, configure its settings, and import a Kerberos keytab file for it.

Before you begin

Before beginning this task, ensure that you have completed all the required configuration tasks that are related to using SPNEGO authentication.

About this task

By default, the SPNEGO authentication is disabled.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the point of contract server profile that you are using in your environment.
4. Click the **Advanced** button. The SOAP Endpoint Security Settings panel opens.
5. Click **SPNEGO Authentication Settings**.
6. In the SPNEGO Authentication Settings panel, select the **Enable SPNEGO Authentication** check box.
7. Complete the fields with the information for your authentication configuration. Refer to the online help for complete descriptions of the fields.
8. Import the Kerberos keytab file to use, 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**.
9. Click **OK**.

Chapter 17. Managing the runtime node

When you installed the runtime and management service component of IBM Tivoli Federated Identity Manager, you configured a node so that it could run as an application on a WebSphere Application Server.

The runtime node management tasks include:

Publishing pages

If you customize an event page or page locale for an event page, you must publish those pages for the customization to take effect.

Before you begin

Complete one or both of the following tasks:

- "Modifying event pages" on page 147
- "Managing page locales" on page 147

About this task

If you did not publish the pages as part of these tasks, you can publish the pages at a later time.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. To publish pages, click the **Publish Pages** button.

Publishing plug-ins

If you have developed the modules for a custom point of contact server or custom identity map module, you must publish the plug-ins for those modules so that you can use them in your IBM Tivoli Federated Identity Manager environment.

Before you begin

Before continuing with this task, ensure that you have developed the appropriate modules for your environment. For more information, see the module topics in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Copy the plug-ins to the /plugins directory where you installed IBM Tivoli Federated Identity Manager. For example, on AIX, the directory is `/opt/IBM/FIM/plugins`.
2. Log on to the console.
3. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
4. Click **Publish Plug-ins**.

What to do next

After publishing the plug-ins, the modules are available for use. New mapping modules are added to the Module Types list.

Viewing custom runtime properties

Custom properties are unique name-value pairs that you can view from the Runtime Custom Properties panel.

Before you begin

Attention: The use of custom properties is an advanced topic. Review the information in the *IBM Tivoli Federated Identity Manager Configuration Guide* before continuing with this procedure.

About this task

To view custom runtime properties:

Procedure

1. Log on to the console .
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. Click **Runtime Custom Properties**. The Runtime Custom Properties panel opens.
4. Select the scope of the custom property, either cell or node, from the **Scope** list. A list of properties at the scope you selected opens.
5. Click **Cancel** when you are done.

Creating a custom property

You can customize the domain configuration by defining a custom property.

Before you begin

Attention: The creation and use of custom properties is an advanced topic. Review the information in the *IBM Tivoli Federated Identity Manager Configuration Guide* before continuing with this procedure.

About this task

The syntax for custom properties is:

```
property_name = property_value
```

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. Click **Runtime Custom Properties**. The Runtime Custom Properties panel opens.
4. Select the scope of the custom property, either cell or node, from the **Scope** list. A list of properties at the scope you selected shows.
5. Click **Create**. A list item is added to the list of properties with the name of **new key** and a value of **new value**.
6. Select the placeholder property.
7. Enter a string in the **Name** field. Do not insert the space character in this field.
8. Enter a string in the **Value** field. Spaces are allowed in this field.
9. Click **Apply** to apply the changes without exiting from the panel, or click **OK** to apply the changes and exit from the panel.

Deleting a custom property

You can delete a custom property that does not meet your needs by its name and value pair.

Before you begin

Attention: The use of custom properties is an advanced topic. Review the information in the*IBM Tivoli Federated Identity Manager Configuration Guide* before continuing with this procedure.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. Click **Runtime Custom Properties**. The Runtime Custom Properties panel opens.
4. Select the scope of the custom property, either cell or node, from the **Scope** list. A list of properties at the scope you selected opens.
5. Select a name and value pair.
6. Click **Delete**. The panel refreshes and the name and value pair is removed from the list of custom properties.
7. 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.

Deploying the runtime node

IBM Tivoli Federated Identity Manager automatically deploys a runtime node. When you make updates to the runtime, such as when you apply a fix pack or language pack to the runtime component, you must manually deploy the runtime node.

About this task

Use the console to deploy the runtime node.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. To deploy the runtime node, click the **Deploy Runtime** button. If the button is inactive, the runtime node is already deployed.

Reloading the configuration

After you make changes to your IBM Tivoli Federated Identity Manager configuration, reload the configuration for the changes to take effect. Typically, after you make a change, you are prompted to load the configuration, however, you are given the option to load it at a later time.

About this task

This task describes how to load a configuration so that configuration changes are recognized.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**. The Runtime Node Management panel opens.
3. To publish pages, click the **Reload Configurations** button.

Removing Tivoli Access Manager configuration for a node

Use the Integrated Solutions Console to remove the configuration for a node before you can remove the runtime application from WebSphere. In a WebSphere cluster environment, you must complete this step for every node before you can remove the runtime.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

About this task

This task removes the IBM Tivoli Federated Identity Manager runtime from the configuration for Tivoli Access Manager for e-business.

The process includes the removal of the application identity (for the IBM Tivoli Federated Identity Manager application) from the Tivoli Access Manager for e-business domain.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**.
2. Select the check box in the **Select** column next to the runtime you want to unconfigure. The **Unconfigure** option is now activated.
3. Click **Unconfigure**. The Runtime Node configuration is removed.
4. Repeat the previous step for every node in the cluster under the following circumstances:
   - If you want to remove the runtime
   - You have deployed IBM Tivoli Federated Identity Manager into a WebSphere cluster environment

   When the process completes, the Runtime Nodes portlet is redrawn.
5. Verify that the check marks have been removed from the **Configured** and **Enabled** fields for the table entry.
6. If you have removed the configuration in preparation for removing the runtime, continue with "Removing a runtime application from WebSphere Application Server."

Removing a runtime application from WebSphere Application Server

Use the undeploy action to remove the IBM Tivoli Federated Identity Manager runtime from the WebSphere server (single server environment) or WebSphere cluster (cluster environment). This step must be taken before a IBM Tivoli Federated Identity Manager domain can be removed.

Before you begin

You must unconfigure the nodes for a runtime before you can remove it. If necessary, see "Removing Tivoli Access Manager configuration for a node" on page 160.

Procedure

1. Click **Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**. The Current Domain portlet and the Runtime Nodes portlet opens.
2. In the Runtime Nodes portlet, click **Remove Runtime**.

   The runtime is removed from the domain. No further input is required. Verify that the **Remove Runtime** button becomes deactivated upon completion of the process. A message shows that indicates that the runtime has been successfully removed from all the nodes.

Chapter 18. About federated identity provisioning

Federated identity provisioning uses an identity provider to conduct federated account provisioning several service provider partners. The trust relationship established between a service provider and identity providers are based on the WS-Provisioning standard.

Provisioning is a key element in identity management for enterprises. Account provisioning is triggered within a company's internal trust domain when there is a change in account status for a user. For example, a new user account must be created when a new employee is hired. Or, user account permissions might need modification when an employee changes job roles.

Enterprise provisioning systems such as IBM Tivoli Identity Manager provide the functions necessary to evaluate role-based provisioning policy, and can create or modify multiple user accounts as appropriate. Management of these accounts includes management of account IDs and passwords to assure secure authenticated communication within the enterprise.

Federated identity provisioning extends these provisioning management activities beyond an internal trust domain. Federated identity provisioning makes it possible to extend local account provisioning at an identity provider to include federated account provisioning out to multiple service provider partners. A service provider, when notified of the federated provisioning request, can perform the local provisioning necessary to supply its service to the specified employee.

Provisioning requests sent between identity providers and service providers must be secure and must be based on open standards. IBM Tivoli Federated Identity Manager satisfies these requirements by providing an implementation of the WS-Provisioning standard.

WS-Provisioning is a specification authored by IBM to provide a Web service interface to communicate provisioning requests and responses. It includes operations for adding, modifying, deleting, and querying provisioning data. It also specifies a notification interface for subscribing to provisioning events. Provisioning data is described using XML and other types of schema. This facilitates the translation of data between different provisioning systems.

WS-Provisioning is part of the Service Oriented Architecture and has been submitted to the Organization for the Advancement of Structured Information Standards (OASIS) Provisioning Service Technical Committee.

IBM Tivoli Federated Identity Manager supports draft version 0.7 of the WS-Provisioning specification. The WS-Provisioning interface is an open standard that is available to other companies that want to develop interoperable provisioning scenarios and systems. The specification is publicly available on the IBM developerWorks Web site:

`http://www.ibm.com/developerworks`

The IBM Tivoli Federated Identity Manager provisioning components are:

- The IBM Tivoli Federated Identity Manager WS-Provisioning Web service
  This service runs as an application on WebSphere Application Server Version 6.0.

- WS-Provisioning connectors

  The connectors run on IBM Tivoli Directory Integrator.

These components are used when a provisioning event is sent from one provider to another provider.

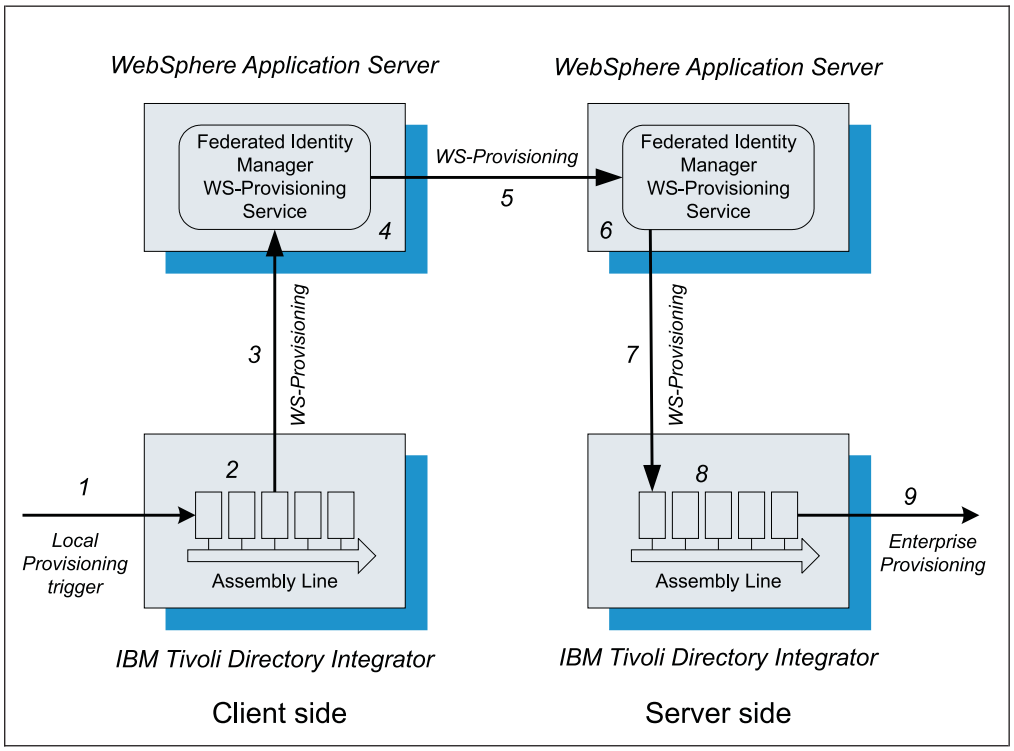


Figure 1. IBM Tivoli Federated Identity Manager federated provisioning process flow

Figure 1 shows the high-level sequence of actions that occur when a client (identity provider) sends a provisioning event to a server (service provider):

1. An external event on the client triggers provisioning events. The event causes an IBM Tivoli Directory Integrator assembly line to be built and executed.
2. The assembly line uses standard IBM Tivoli Directory Integrator connectors to collect the data necessary to form a WS-Provisioning message.

   IBM Tivoli Directory Integrator provides a set of standard connectors that support the sending and receiving of Web services messages. IBM Tivoli Federated Identity Manager configures these connectors by using the WS-Provisioning WSDL.
3. The IBM Tivoli Directory Integrator connectors send a WS-Provisioning SOAP message to the IBM Tivoli Federated Identity Manager WS-Provisioning service.
4. The IBM Tivoli Federated Identity ManagerWS-Provisioning service is a proxy server for WS-Provisioning messages that allows the addition of WS-Security policy for securing the Web services messages between partners.

   The addition of WS-Security policy to the WS-Provisioning service is optional, but typically is included in real-world deployments.
5. The WS-Provisioning service on the client side sends the message to the WS-Provisioning service on the server side.
6. The WS-Provisioning service on the server side receives the message and processes it.

The processing includes handling of WS-Security information. The IBM Tivoli Federated Identity Manager WS-Provisioning service can be configured to use either WebSphere Application Server WS-Security handling, or to use IBM Tivoli Federated Identity Manager Web services security management (not shown here) to authorize the WS-Provisioning request.

7. The WS-Provisioning service sends the message to a configured connector on the local IBM Tivoli Directory Integrator.
8. The IBM Tivoli Directory Integrator connector receives the WS-Provisioning message and starts a configured assembly line.

   The assembly line collects any local data that is required.
9. The assembly line initiates local provisioning.

   This local provisioning can include the use of an enterprise provisioning system that uses Directory Services Markup Language Version 2 (DSMLv2), such as IBM Tivoli Identity Manager.

Federated provisioning components

The IBM Tivoli Federated Identity Manager provisioning solution consists of a WS-Provisioning service and a set of functional components for an IBM Tivoli Directory Integrator assembly line.

The following sections describe the components and how they are used:

- "WS-Provisioning service"
- "Transaction sequence for the assembly line and provisioning service" on page 169

WS-Provisioning service

The IBM Tivoli Federated Identity Manager WS-Provisioning *provisioning service* is a Java 2 Enterprise Edition (J2EE) application. This application routes WS-Provisioning messages to a destination provisioning service. It also acts as a security proxy for incoming and outgoing WS-Provisioning SOAP messages.

The provisioning service offloads the cost of securing messages from the client and server WS-Provisioning endpoints that uses the proxy. The provisioning service uses WS-Security and WS-Trust to secure WS-Provisioning messages. The WS-Provisioning specification does not require that messages be secured. However, real-world deployments combine WS-Provisioning with WS-Security and WS-Trust to ensure the security and integrity of the messages

- For outgoing messages, the application uses WebSphere to add WS-Security to the SOAP message and then proxies the secured message on to its destination. An example of a destination is another provisioning service.
- For incoming messages, the application uses either WebSphere or the IBM Tivoli Federated Identity Manager Web Services Security Manager (WSSM) to authenticate the user by using the WS-Security information in the SOAP message.

  When the user has been authenticated, the application removes the WS-Security information and proxies (sends) the SOAP request on to its destination. The destination is a configured WS-Provisioning endpoint. An example of a destination is an IBM Tivoli Directory Integrator assembly line.

Table 17 on page 166 lists the IBM Tivoli Federated Identity Manager support for the WS-Provisioning interfaces.

Table 17. Support for WS-Provisioning interfaces

WS-Provisioning interface	Status
Provisioning interface	Supported
Notification interface	Not supported
Notification listener interface	

The implementation of the provisioning interface consists of the following functions:

- Receive a provisioning request.
- Forward the request to the configured target provisioning service endpoint.
- Receive the provisioning response and return it to the sender of the original provisioning request.

WS-Provisioning operations

View the lists of WS-Provisioning operations that are supported and not support by IBM Tivoli Federated Identity Manager.

The IBM Tivoli Federated Identity Manager provisioning service supports the following WS-Provisioning operations:

- listTargets
- fetchTargets
- listProvisionedItems
- fetchProvisionedItems
- listProvisionedLifecycle
- listRequestStatus
- cancelRequest
- provision
- deprovision
- modifyProvisionedState
- modifyProvisionedParameters

The following WS-Provisioning operations are not supported:

- subscribe
- unsubscribe
- notify

For more information on WS-Provisioning operations, review the draft WS-Provisioning specification:

`http://www.ibm.com/developerworks/webservices/library/ws-provis/`

WS-Security

Learn why provisioning requests and responses must be encrypted and signed, and the ways to apply WS-Security to the provisioning service application.

In a federated provisioning deployment, provisioning requests and responses are exchanged across a distributed network environment. These exchanges can include messages sent between business enterprises. The provisioning requests and responses typically contain sensitive data and therefore are encrypted and signed.

Provisioning requests and responses include a security token to authenticate the originator of the request and to make authorization decisions. The request SOAP body and the authentication security token are signed, and the SOAP body are encrypted. The response SOAP body is also signed and encrypted.

The provisioning service employs the Web services security infrastructure that is provided by the J2EE application server, such as WebSphere Application Server and configured by using deployment descriptors. This security and its configuration are transparent to the provisioning service application, with no security-specific code required in the application.

One method of encrypting and signing provisioning messages is to use the Secure Socket Layers (SSL) protocol. This method secures messages at the transport layer. Another method is to use the WS-Security standard for message-based security.

The IBM Tivoli Federated Identity Manager provisioning service does not, by default, contain WS-Security configuration. You can use WebSphere Application Server or IBM Tivoli Federated Identity Manager Web services security manager to add WS-Security configuration.

The configuration of WS-Security is dependent on factors specific to each deployment. These factors include:

- The role of the provisioning service instance.

  The role of the provisioning service is determined by whether it is responsible for *sending* provisioning requests or *receiving* provisioning requests. The *client* role sends provisioning requests. The *server* role receives provisioning requests and sends responses. In a federated provisioning deployment that includes federated single sign-on, the client side is the *identity provider* and the server side is the *service provider*.
- The specific business policy security requirements for signing and encryption.

  Implementation of these requirements includes the definition and configuration of deployment-specific encryption keys.

Each instance of the provisioning service must be secured separately.

There are two ways to apply WS-Security to the provisioning service application:

- Use the WS-Security methods supplied by WebSphere Application Server .

  The methods can be used on the client side, the server side, or both sides.

  The WebSphere functionality, when used on the server side, includes basic token validation modules but does not include use of Tivoli Access Manager for authorization.
- Use the IBM Tivoli Federated Identity Manager Web services security manager

  This method can be used *on the server side only.*

  The IBM Tivoli Federated Identity Manager Web services security manager (WSSM) contains a component called the *WSSM token consumer*. The WSSM token consumer provides a callout to the IBM Tivoli Federated Identity Manager trust service. The callout is used for token validation and for web services authorization. The trust service uses Tivoli Access Manager to complete the authorization.

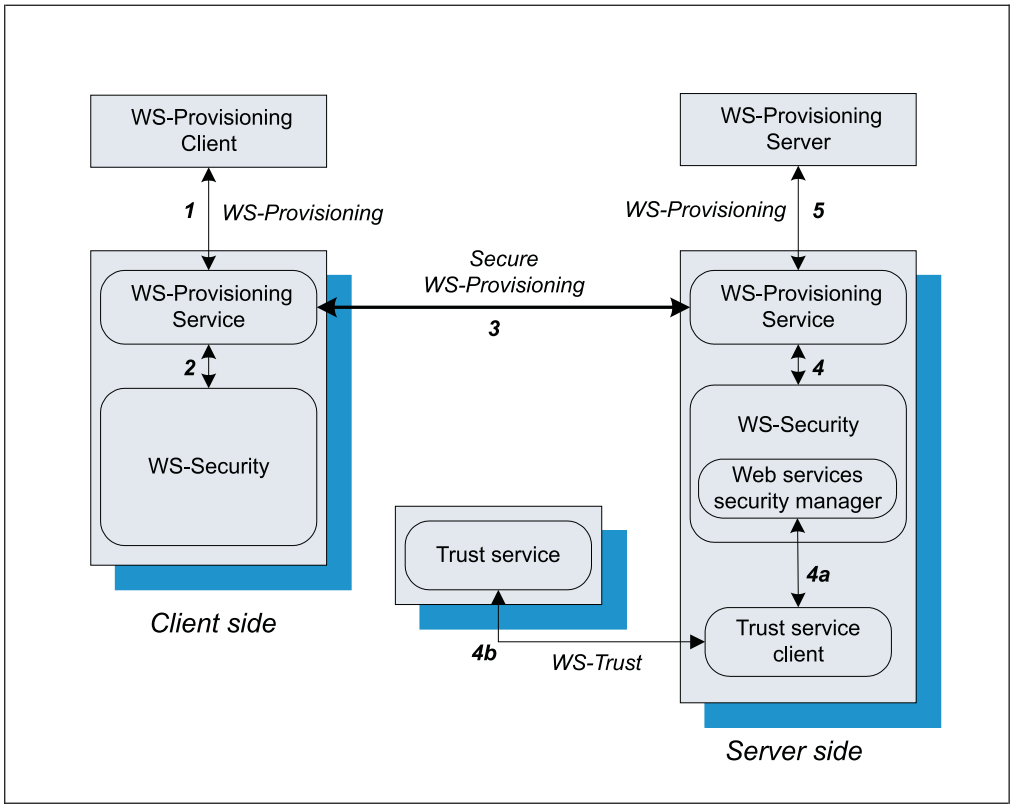


Figure 2. WS-Security use in the flow of WS-Provisioning messages

Figure 2 shows the message flow when securing WS-Provisioning messages:

1. The WS-Provisioning client sends a WS-Provisioning message to the provisioning service. The message is not secured with WS-Security.

   An example of a WS-Provisioning client is the IBM Tivoli Directory Integrator assembly line.
2. The provisioning service adds WS-Security based on the configuration of the deployment descriptors.

   This is the *client* or *sending* side of the provisioning deployment. On this side, WebSphere Application Server is used to add WS-Security.
3. The secured message is sent to the provisioning service on the *server* or *receiving* side of the deployment.
4. The provisioning service processes the message according to the WS-Security configuration.

   The processing steps vary depending on the WS-Security configuration used.

   - When WebSphere Application Server has been used to configure WS-Security, the message is processed based on the coarse-grained authorization provided by WebSphere Application Server .
   - When IBM Tivoli Federated Identity Manager Web services security manager has been used, finer-grained authorization can be achieved through the use of Tivoli Access Manager .

     a. The IBM Tivoli Federated Identity Manager message processing includes a call to a trust client (4a).

     b. The trust client calls to the IBM Tivoli Federated Identity Manager trust service (4b) to obtain authorization information for the request.

The trust service calls out to Tivoli Access Manager (not shown) for the authorization decision.

5. The WS-Provisioning message is sent to the WS-Provisioning server.

   An example of a WS-Provisioning server is an IBM Tivoli Directory Integrator assembly line. The assembly line can process the message and, for example, deliver it to a local provisioning service such as IBM Tivoli Identity Manager.

Note: For more information about the Web services security manager, see the IBM Tivoli Federated Identity Manager information center at: http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html

Provisioning service deployment

The provisioning service can receive requests from multiple WS-Provisioning clients, but it can forward requests to only a single target provisioning Web service.

The handling of multiple requests is not supported because there is no standard way defined for the provisioning service to determine which target is to receive a request. As a result, only one proxy URL is configured.

When an identity provider wants to provision multiple service providers, the identity provider must deploy multiple instances of the provisioning service. This deployment is accomplished through the IBM Tivoli Federated Identity Manager runtime deployment. Similarly, when the service provider has multiple provisioning systems then it must deploy multiple instances of the provisioning service.

There is no management interface to the provisioning service. The proxy URL is the only configuration parameter. The proxy URL is configured by setting a global parameter in the IBM Tivoli Federated Identity Manager runtime custom properties.

Note: See "Configuring the client side" on page 193 for instructions on how to configure the proxy URL.

Transaction sequence for the assembly line and provisioning service

IBM Tivoli Federated Identity Manager uses IBM Tivoli Directory Integrator to support enterprise-specific provisioning systems.

IBM Tivoli Directory Integrator is a tool for integrating disparate data sources. A data source is accessed by using an IBM Tivoli Directory Integrator connector that is specific to the type of data source. For example, an LDAP *connector* is used by IBM Tivoli Directory Integrator to access LDAP directories.

IBM Tivoli Directory Integrator supports *functional components*. A functional component is a unit of logic that is similar to a connector but without the built-in logic that is required to interface with a specific type of data source.

IBM Tivoli Directory Integrator connectors and functional components are configured into an *assembly line* in order to receive data, manipulate the data, and transport the data. Connectors and functional components present a common interface to assembly lines so that data can be processed in a uniform manner.

Assembly lines support the mapping of data between connectors and functional components. This means that the outputs from one connector become the inputs to another connector. Complex mapping, including the use of JavaScript, is also supported.

IBM Tivoli Directory Integrator event handlers or connectors, when running in server mode, can be triggered by an external event. When triggered, the handlers or connectors start an assembly line. For example, the IBM Tivoli Directory Integrator LDAP event handler triggers a configured assembly line when LDAP directory changes match a previously established selection criteria.

Note: For more information, see the IBM Tivoli Directory Integrator documentation on the Tivoli information center:

`http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp`

The IBM Tivoli Federated Identity Manager implementation of provisioning uses an assembly line on both the client side and server side. Both assembly lines are used when completing a provisioning request. The following list describes the sequence of assembly line events when a client submits a provisioning request.

1. An external event causes an IBM Tivoli Directory Integrator event handler to be notified.

   In the demonstration scenario, the external event is an update to the IBM Directory Server, as flagged by an LDAP Change Log entry.
2. The event handler starts the assembly line (Figure 3 on page 171). The event handler passes through any information that is required for the operation. The assembly line completes the following steps:
   a. The Customer Connector takes the configured inputs, builds a WS-Provisioning Java object, and passes it to the Axis JavaToSoap functional component.
   b. The Axis JavaToSoap functional component builds a WS-Provisioning SOAP request and passes it to the InvokeSoapWS functional component.
   c. The InvokeSoapWS functional component produces a WS-Provisioning request and sends it to the IBM Tivoli Federated Identity Manager provisioning service.

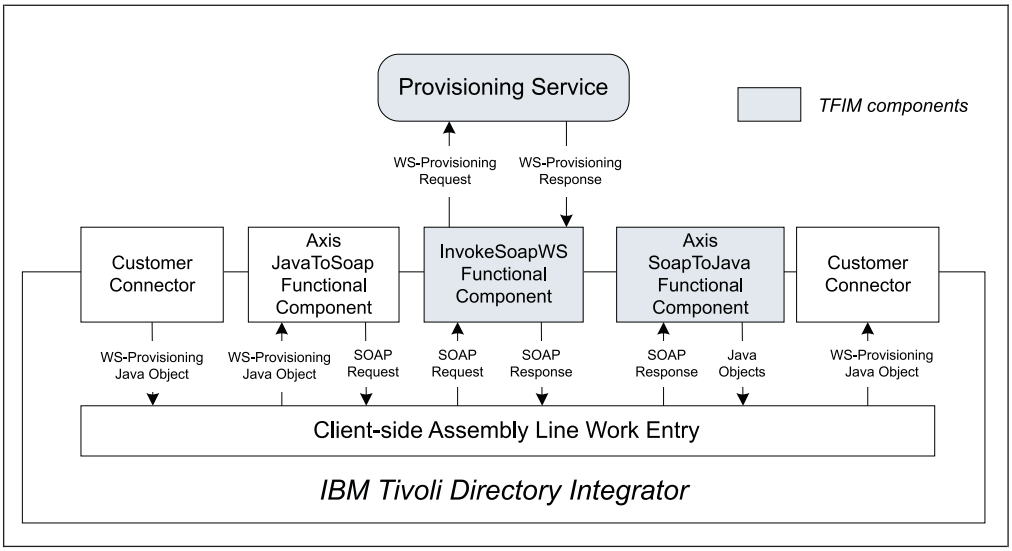


Figure 3. Client-side (identity provider) assembly line

3. The client-side (identity provider) provisioning service optionally adds WS-Security to the SOAP request. The provisioning service sends the SOAP message to the server-side (service provider) provisioning service.
4. The service provider provisioning service receives the SOAP request. The provisioning service:
   a. Authenticates the caller by using WS-Security information.

      This authentication can include use of the WSSM token consumer within the WS-Security configuration. Use of the WSSM token consumer allows the IBM Tivoli Federated Identity Manager trust service to authenticate the caller and to use Tivoli Access Manager to authorize the request.
   b. Removes the WS-Security information and proxies the request to the IBM Tivoli Directory Integrator entry point.
5. The server-side entry point for IBM Tivoli Directory Integrator starts the assembly line and passes through the information required for the provisioning operation. The assembly line (Figure 4 on page 172) completes the following steps:
   a. The WS-Receiver server connector passes the SOAP request to the Axis SoapToJava functional component.
   b. The Axis SoapToJava functional component generates the WS-Provisioning request Object and passes it to the Customer assembly line connectors.
   c. The Customer assembly line connectors run the provisioning operation and return a WS-provisioning response object to the Axis JavaToSoap functional component.
   d. The Axis JavaToSoap functional component packages the WS-provisioning response object as a SOAP response and passes it to the WS-Receiver server connector.
   e. The WS-Receiver server connector sends the response to the server-side provisioning service.

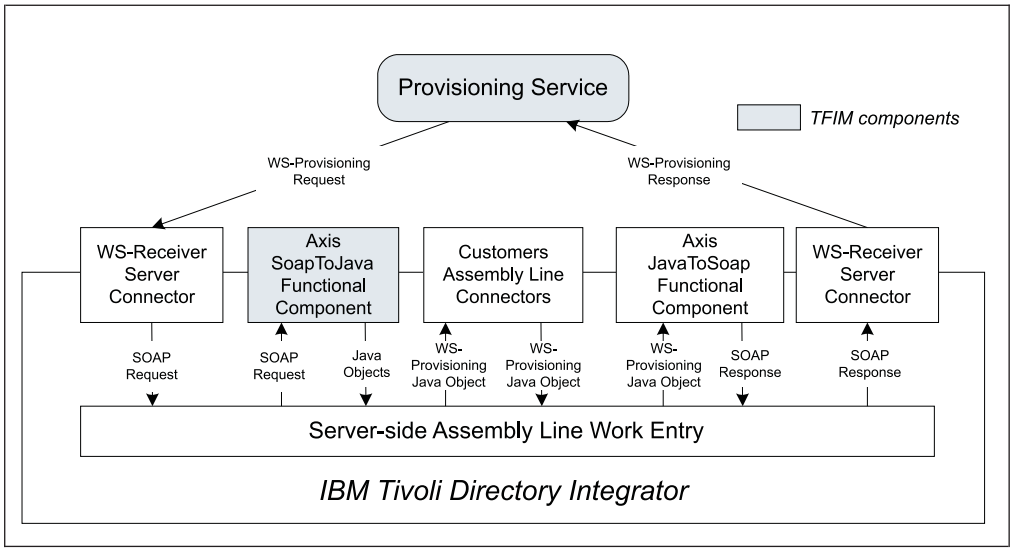


Figure 4. Server-side (service provider) assembly line

6. The server-side provisioning service adds WS-Security to the response and sends it to the client-side provisioning service.
7. The client-side provisioning service receives the SOAP response from the service provider.
8. The provisioning service authenticates the response using WS-Security headers. When the authentication completes, the provisioning service removes the WS-Security headers from the SOAP request and returns the WS-Provisioning response to the assembly line (Figure 3 on page 171). The assembly line completes the following steps:
   a. The InvokeSoapWS functional component receives the WS-Provisioning response, produces a corresponding SOAP response, and passes the SOAP response to the Axis SoapToJava functional component.
   b. The Axis SoapToJava functional component converts the SOAP response to a WS-Provisioning response object, and sends it to the Customer Connector.

WS-Provisioning demonstration scenario

The IBM Tivoli Federated Identity Manager provisioning component includes a simple customer-provisioning scenario. The scenario provides additional code that has been developed to work with the IBM Tivoli Federated Identity Manager WS-Provisioning components.

The sample scenario uses the fictional entities MyEmployer and BenefitsCompany as examples of the provider roles within a IBM Tivoli Federated Identity Manager single sign-on federation. MyEmployer is an identity provider, and BenefitsCompany is a service provider.

The provisioning target for the WS-Provisioning scenario is Tivoli Access Manager running on the BenefitsCompany server. There is a one-to-one mapping between Tivoli Access Manager user accounts at MyEmployer and Tivoli Access Manager user accounts at BenefitsCompany. That is, for every account at MyEmployer there is a corresponding Tivoli Access Manager account at BenefitsCompany. These accounts share the same user name.

Figure 5 shows XML code that represents the ProvisioningTarget for Tivoli Access Manager at BenefitsCompany.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:ProvisioningTarget xmlns:core="urn:ibm:names:ws:provisioning:0.1:core"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ibm:names:ws:provisioning:0.1:core core.xsd
 http://www.w3.org/XML/1998/namespace xml.xsd ">

  <core:identifier name="BenefitsCompany TAM"/>

</core:ProvisioningTarget>
```

Figure 5. XML code defining the ProvisioningTarget at BenefitsCompany

Figure 6 shows the XML schema for defining a user in the Tivoli Access Manager target:

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://www.ibm.com/TAMUser"
        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core"
        xmlns:tamuser="http://www.ibm.com/TAMUser"
        xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

        <import namespace="urn:ibm:names:ws:provisioning:0.1:core"
                schemaLocation="core.xsd" />

        <complexType name="TAMUserType">
                <sequence>
                        <element name="ip_identifier" type="string" />
                        <element name="username" type="string" minOccurs="0" />
                        <element name="password" type="string" minOccurs="0" />
                        <element name="email" type="string" minOccurs="0" />
                </sequence>
        </complexType>

        <element name="TAMUser" type="tamuser:TAMUserType" />

</schema>
```

Figure 6. XML schema for defining a Tivoli Access Manager user

The element `ip_identifier` is used to uniquely identify the Tivoli Access Manager user. This element is used because the IBM Tivoli Directory Integrator event handler that detects modifications to LDAP at the identity provider does not always have access to the Tivoli Access Manager user name. In particular, when an LDAP delete operation occurs, only the Distinguished Name (DN) of the object being deleted is made available to the LDAP event handler. The `ip_identifier` element is the base-64 encoded LDAP DN of the user's ePerson record.

The service provider treats this identifier as an opaque string and stores it with the user's record when creating a new user. When a delete or modification occurs, the service provider uses this identifier to perform a lookup for the user, and operates on the appropriate user object.

The demonstration scenario exercises a subset of the supported WS-Provisioning operations. The operations are handled synchronously and are triggered by an IBM

Tivoli Directory Integrator connector at MyEmployer. The connector monitors the LDAP change log of the Tivoli Access Manager user registry at MyEmployer.

The WS-Provisioning operations used in the demonstration scenario are:

- "provision()"
- "deprovision()" on page 175
- "modifyProvisionedState()" on page 176
- "modifyProvisionedParameters()" on page 177

provision()

A WS-Provisioning operation that allows the originally created accounts from the identity provider to be created in the service provider as well.

This operation creates a new Tivoli Access Manager user account at BenefitsCompany, based upon the addition of an account at MyEmployer. The Tivoli Access Manager **pdadmin** operations at MyEmployer that result in a provision() call are:

```
user create
user import
```

Figure 7 shows an example SOAP message for a sample provision() request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
                <api:ProvisionRequest
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:target name="BenefitsCompany TAM" />
                        <api:parameters>
                            <tamuser:TAMUser
                                xmlns:tamuser="http://www.ibm.com/TAMUser">
                             <tamuser:ip_identifier>opaque_id
                                </tamuser:ip_identifier>
                             <tamuser:username>bmarley</tamuser:username>
                           </tamuser:TAMUser>
                        </api:parameters>
                </api:ProvisionRequest>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 7. Example SOAP message for provision() request

Figure 8 on page 175 shows an example SOAP message for a sample provision response.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:ProvisionResponse
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:status>
                                <core:code>success</core:code>
                        </api:status>
                        <api:item>
                            <core:identifier name="opaque_string" />
                            <core:target name="BenefitsCompany TAM" />
                            <core:state>created</core:state>
                            <core:parameters>
                                <tamuser:TAMUser
                                   xmlns:tamuser="http://www.ibm.com/TAMUser">
                                   <tamuser:ip_identifier>opaque_string
                                      </tamuser:ip_identifier>
                                   <tamuser:username>bmarley</tamuser:username>
                                 </tamuser:TAMUser>
                                </core:parameters>
                        </api:item>
                </api:ProvisionResponse>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 8. Example SOAP message for provision() response

deprovision()

A WS-Provisioning operation that mimics the deletion of an account at the service provider in the identity provider.

This operation deletes a Tivoli Access Manager account at BenefitsCompany based on the deletion of a Tivoli Access Manager account at MyEmployer. The only **pdadmin** operations at MyEmployer that result in a deprovision() call are:

`user delete -registry` *`name_of_user`*

Figure 9 on page 176 shows an example SOAP message for a sample deprovision() request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:DeprovisionRequest
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:item>
                                <core:identifier name="opaque_string" />
                                <core:target name="BenefitsCompany TAM" />
       </api:item>
                </api:DeprovisionRequest>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 9. Example SOAP message for deprovision() request

Figure 10 shows an example SOAP message for a sample deprovision() response.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:DeprovisionResponse
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:status>
                                <core:code>success</core:code>
                        </api:status>
                </api:DeprovisionResponse>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 10. Example SOAP message for deprovision() response

modifyProvisionedState()

The account at the service provider is enabled or disabled depending on the command done at the identity provider.

This operation enables or disables the corresponding Tivoli Access Manager user account at BenefitsCompany based on Tivoli Access Manager user commands performed at MyEmployer. Sample Tivoli Access Manager **pdadmin** account command:

`user modify` *`name_of_user`* `account-valid [ yes | no ]`

Figure 11 on page 177 shows an example SOAP message for a sample modifyProvisionedState() request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:ModifyProvisionedStateRequest
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:item>
                                <core:identifier name="opaque_string" />
                                <core:target name="BenefitsCompany TAM" />
                        </api:item>
                        <api:state>suspended</api:state>
                </api:ModifyProvisionedStateRequest>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 11. Example SOAP message for modifyProvisionedState() request

Figure 12 shows an example SOAP message for a sample modifyProvisionedState() response.

Note: The response contains information about the new state of the provisioned item, but does not contain all of the parameters for the item.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:ModifyProvisionedStateResponse
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:status>
                                <core:code>success</core:code>
                        </api:status>
                        <api:item>
                                <core:identifier name="opaque_string" />
                                <core:target name="BenefitsCompany TAM" />
                                <core:state>suspended</core:state>
                        </api:item>
                </api:ModifyProvisionedStateResponse>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 12. Example SOAP message for modifyProvisionedState() response

modifyProvisionedParameters()

A WS-Provisioning operation that is responsible for updating an attribute at the service provider when changes are made to that same attribute at the identity provider.

This operation is used to update the **emailAddress** attribute of a user at BenefitsCompany when that attribute changes at **MyEmployer**. For this scenario, no other attribute changes are propagated from **MyEmployer** to **BenefitsCompany**.

Figure 13 shows an example SOAP message for a sample modifyProvisionedParameters() request.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:ModifyProvisionedParametersRequest
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:item>
                                <core:identifier name="opaque_string" />
                                <core:target name="BenefitsCompany TAM" />
                        </api:item>
                        <api:modification>
                             <api:operation>replace</api:operation>
                             <api:selector>
                                    <core:select>/tamuser:TAMUser/email
                                          </core:select>
                                    <core:namespace prefix="tamuser"
                                          uri="http://www.ibm.com/TAMUser" />
                                     </core:select>
                              </api:selector>
                              <api:parameters>
                                  <tamuser:email
                                     xmlns:tamuser="http://www.ibm.com/TAMUser">
                                            bob_marley@myemployer.com
                                     </tamuser:email>
                                </api:parameters>
                        </api:modification>
                </api:ModifyProvisionedParametersRequest>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 13. Example SOAP message for modifyProvisionedParameters() request

Figure 14 on page 179 shows an example SOAP message for a sample modifyProvisionedParameters() response.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>

                <api:ModifyProvisionedParametersResponse
                        xmlns:api="urn:ibm:names:ws:provisioning:0.1:api"
                        xmlns:core="urn:ibm:names:ws:provisioning:0.1:core">
                        <api:requestStatus>
                                <core:code>success</core:code>
                        </api:requestStatus>
                        <api:modificationStatus>
                                <core:status>
                                        <core:code>success</core:code>
                                </core:status>
                        </api:modificationStatus>
                </api:ModifyProvisionedParametersResponse>

        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 14. Example SOAP message for modifyProvisionedParameters() response

Chapter 19. Configuring provisioning

Configuration of IBM Tivoli Federated Identity Manager provisioning includes integration with IBM Tivoli Directory Integrator and configuration of the provisioning service.

Securing of the provisioning service is optional, but is typically included in the configuration steps.

Complete the instructions in each of the following topics:

- "Deploying the IBM Tivoli Directory Integrator file"
- "Configuring the provisioning service" on page 182
- "Securing the provisioning service" on page 183

Deploying the IBM Tivoli Directory Integrator file

IBM Tivoli Federated Identity Manager provides a WS-Provisioning JAR file for use by IBM Tivoli Directory Integrator.

About this task

The file must be copied from the default IBM Tivoli Federated Identity Manager directory to the IBM Tivoli Directory Integrator installation directory.

The IBM Tivoli Federated Identity Manager runtime component is installed on a system that also hosts WebSphere Application Server. IBM Tivoli Directory Integrator can be installed on that same system or on a remote system. This means that you must create either a *local* copy or a *remote* copy of the JAR file, depending on your deployment.

You can create a local copy in the following deployment environments:

- When IBM Tivoli Directory Integrator is to be used in a WebSphere Application Server single-server deployment, and is installed on the *same* system as the WebSphere Application Server.
- When IBM Tivoli Directory Integrator is to be used in a WebSphere Application Server cluster deployment, and is installed on the *same* system as the Deployment Manager.

You must create a remote copy in the following deployment environments:

- When IBM Tivoli Directory Integrator is to be used in a WebSphere Application Server single-server deployment, and is installed on a *different* system than the WebSphere Application Server.
- When IBM Tivoli Directory Integrator is to be used in a WebSphere Application Server cluster deployment, and is installed on a *different* system than the Deployment Manager.

  Note: The IBM Tivoli Directory Integrator can be located on one of the cluster nodes, or on a host system that is not configured into the cluster.

Procedure

1. Complete the file copy:
   - UNIX

     Copy:

     ```
     /opt/IBM/FIM/provisioning/idijars/itfim-provisioning.jar
     ```

     into this directory:

     ```
     /opt/IBM/IBMDirectoryIntegrator/jars
     ```

   - Windows

     Copy:

     ```
     C:\Program Files\opt\IBM\FIM\provisioning\idijars\
        itfim-provisioning.jar
     ```

     into this directory:

     ```
     C:\Program Files\opt\IBM\IBMDirectoryIntegrator\jars
     ```

2. Continue with "Configuring the provisioning service."

Configuring the provisioning service

You must configure the custom properties of the provisioning service for both client and server sides.

Procedure

1. If you are configuring the provisioning service into a WebSphere Application Server *cluster* environment, you must deploy the provisioning service EAR file.

   Note: If you are configuring the provisioning service into a WebSphere Application Server *single-server* environment, skip this step. Continue with step 2.

   The IBM Tivoli Federated Identity Manager provisioning service EAR file is installed as part of the IBM Tivoli Federated Identity Manager runtime component.

   When you install the runtime component onto a WebSphere Application Server ND (Network Deployment) Deployment Manager, as part of a cluster environment, the provisioning service file is also installed on the Deployment Manager. This file in installed as:

   ```
   provisioning/itfim-provisioning.ear
   ```

   When you *deploy* the runtime component, the IBM Tivoli Federated Identity Manager runtime files are programmatically pushed from the Deployment Manager out to each node in the cluster. However, the provisioning service EAR file is *not* pushed to each node. You must manually copy the provisioning file to each of the target nodes.

   For each node:

   a. Create the directory `provisioning` under the IBM Tivoli Federated Identity Manager runtime installation directory.

   b. Copy `itfim-provisioning.ear` from the Deployment Manager to the `provisioning` directory on the node.

2. Specify the proxy URL configuration.

   The proxy URL is configured using the *custom properties* feature of the IBM Tivoli Federated Identity Manager runtime. Use the IBM Tivoli Federated Identity Manager console to specify custom properties. The default custom runtime properties file contains a placeholder entry for the proxy service URL. The default setting is:

```
provisioning.proxyDestinationURL =
   http://your.provisioning.endpoint:9999/wsp/wspservice
```

- On the client side, you must change this value to point to the partner WS-Provisioning service.
- On the server side, you must change this value to point to the local IBM Tivoli Directory Integrator service.

a. Log on to the IBM Tivoli Federated Identity Manager administration console.

b. Click **IBM Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**.

   The Runtime Nodes portlet opens.

c. Click **Runtime Custom Properties**.

   The Runtime Custom Properties portlet opens.

d. Select the table row that contains the Name entry **provisioning.proxyDestinationURL**.

e. Replace the default entry in the Value column with the value of your provisioning endpoint.

   - Client side:

     For example, when the host system of the provisioning service is `sp.example.com`, and WebSphere Application Server listens on port `9080`, an example value is:

     ```
     http://sp.example.com:9080/wsp/wspservice
     ```

   - Server side:

     For example, when the host system of the local IBM Tivoli Directory Integrator service is `sp.example.com`, and the IBM Tivoli Directory Integrator assembly line processes requests on port 8888, an example value is:

     ```
     http://sp.example.com:8888/wsp/wspservice
     ```

f. Click **OK**.

Results

Note: The provisioning service does not require that a IBM Tivoli Federated Identity Manager single sign-on federation be created and configured. However, in some cases, you might want to also configure a IBM Tivoli Federated Identity Manager single sign-on federation. The provisioning service can be used with federated single sign-on to enable federated user accounts to be provisioned.

Securing the provisioning service

You must modify the provisioning service Web module that is distributed as part of the IBM Tivoli Federated Identity Manager runtime component application to secure the provisioning service.

About this task

WS-Security configuration is used to protect messages between the client side provisioning service and the server-side provisioning service. You can use WebSphere Application Server to add WS-Security on both the client side and server side.

(Optional) On the server side only, you can use the IBM Tivoli Federated Identity Manager Web services security manager (WSSM) to add WS-Security. When using WSSM, you can configure the WSSM token consumer to provide a callout to the IBM Tivoli Federated Identity Manager trust service for authorization and security token validation.

You must then redeploy the IBM Tivoli Federated Identity Manager runtime application after the provisioning service Web module has been modified. The deployment steps vary depending on whether you have a WebSphere cluster environment or a stand-alone WebSphere Application Server.

Procedure

1. Use a development environment, such as Rational® Application Developer, to update the deployment descriptors in the IBM Tivoli Federated Identity Manager runtime component. Tasks:
   a. Import the runtime EAR file into a development environment, in preparation for modifying the file contents.
   b. Update the Web Services Security deployment descriptors for the provisioning service.
   c. Export the modified descriptors (as the development project) to a new IBM Tivoli Federated Identity Manager EAR file, and copy it to the appropriate location
2. Use the IBM Tivoli Federated Identity Manager administration console to deploy and configure the updated IBM Tivoli Federated Identity Manager runtime component.
   a. Unconfigure and deploy any existing runtime component.
   b. Deploy and configure the updated runtime component.

Results

Complete the instructions in the following topics:

1. "Adding WS-Security to the provisioning runtime component"
2. "Deploying the updated provisioning runtime component" on page 187

Adding WS-Security to the provisioning runtime component

You must complete several tasks to update the deployment descriptors in the IBM Tivoli Federated Identity Manager runtime component.

Complete the following instructions:

- "Importing the runtime component into a development toolkit"
- "Adding WS-Security configuration to the provisioning service" on page 185
- "Exporting the IBM Tivoli Federated Identity Manager project" on page 187

Importing the runtime component into a development toolkit

You can use either Rational Application Developer or WebSphere Application Server Toolkit to modify the IBM Tivoli Federated Identity Manager runtime component.

About this task

The following instructions describe the use of Rational Application Developer. The steps required for WebSphere Application Server Toolkit are similar.

Procedure

1. Start Rational Application Developer.
2. Select **File → Import**. The Import window opens. You are prompted to select an external EAR file for import into an Enterprise Application project.
3. Select **EAR file**.
4. Click **Next**.
5. Browse to the IBM Tivoli Federated Identity Manager runtime file:
   - UNIX

     `/opt/IBM/FIM/pkg/release/itfim.ear`
   - Windows

     `C:\Program Files\opt\FIM\pkg\release\itfim.ear`
6. Import the file.
7. Accept the default settings.
8. Click **Finish**.
9. When prompted, switch to J2EE Perspective. Ignore any messages in the Problems window related to building the product.
10. Continue to "Adding WS-Security configuration to the provisioning service."

Adding WS-Security configuration to the provisioning service

You must add the WS-Security configuration to the provisioning service to update the runtime component.

About this task

This topic continues from the instructions in "Importing the runtime component into a development toolkit" on page 184.

Continuing in the Rational Application Developer, complete the following steps in this procedure:

Procedure

1. Expand the **Web Services** folder.

   The folder contains two entries: **Services** and **Clients**.
2. Choose one:
   - When you are deploying provisioning on the client side, use the **Clients** section. Edit the **ProvisioningServiceWeb: service/ProvisioningService** entry. Double click the entry to edit it.
   - When you are deploying provisioning on the server side, use the **Services** section. Edit the **ProvisioningService** entry. Double click the entry to edit it.
3. Use either the WebSphere documentation or the *IBM Tivoli Federated Identity Manager Web Services Security Manager Guide* to add appropriate WS-Security bindings and extensions configuration items for the Web services deployment descriptors. .
   - To use the WebSphere documentation, continue with step 4.
   - To use the *IBM Tivoli Federated Identity Manager Web Services Security Manager Guide*, continue with step 5 on page 186.
4. Access the **WebSphere Application Server information center**

   `http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp.`

a. In the left panel, select **WebSphere Application Server Network Deployment, Version 6.0.x**.
b. In the left panel, select **How do I? Securing applications and their environments.**
c. In the right panel, scroll down to the section **Use Web services security (WS-Security)**.
d. Select the **Documentation** link.
e. Review the topics displayed in the right panel, and complete the necessary steps to secure the provisioning service application.
f. Continue with step 6.

5. Access the **IBM Tivoli Federated Identity Manager information center** at http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html
   a. On the Welcome page (right panel), select the **Additional product documentation** link.
   b. Select the link for the *IBM Tivoli Federated Identity Manager Web Services Security Manager Guide*.
   c. Follow the instructions for installing and configuring the IBM Tivoli Federated Identity Manager Web services security manager component.
   d. Complete the topic *Enabling Web services security management*.
   e. Complete the topic *Enabling application security*.
   f. Complete the topic *Enabling global security in WebSphere Application Server*.
6. Configure the keystore files.

   Note: This step applies only within a WebSphere cluster environment. If you are deploying into a WebSphere single-server environment, skip this step and continue with step 7 on page 187.

   When you reference keystore files for the encryption and the signing of keys, each keystore file must be copied to each WebSphere Application Server node where the IBM Tivoli Federated Identity Manager runtime component is deployed.

   - The WebSphere WS-Security examples demonstrate use of the `${USER_INSTALL_ROOT}` variable as a prefix to the keystore path. When WebSphere is installed in the default location, the value of this variable is `/opt/IBM/WebSphere/AppServer`.
   - The IBM Tivoli Federated Identity Manager Web services security manager documentation shows an example where the keystore file name is used without a preceding path variable. When using this type of naming, the default directory for locating the file is the profile-specific subdirectory under WebSphere.

   For example:

   - *If* your keystore file is named `wssm_client.jks`
   - *And* the name `wssm_client.jks` is entered as the keystore path, without a prepended WebSphere variable or path
   - *And* the Application Server profile is `AppSrv01`
   - *Then* the `wssm_client.jks` file is copied to:
     `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01`

   Note: This file must be manually copied to *every* node in the cluster.

7. Continue to "Exporting the IBM Tivoli Federated Identity Manager project."

Exporting the IBM Tivoli Federated Identity Manager project

Export the IBM Tivoli Federated Identity Manager project using Rational Application Developer or the WebSphere Application Server Toolkit.

About this task

This topic continues from the instructions in "Adding WS-Security configuration to the provisioning service" on page 185.

After you have completed the configuration of WS-Security, export the modified IBM Tivoli Federated Identity Manager project. Use either Rational Application Developer or the WebSphere Application Server Toolkit.

Procedure

1. In the development toolkit, open the **Enterprise Applications** folder in the J2EE Perspective view.
2. Locate the project and right-click on the project.

   For example, you would right-click on the **itfim** project. A menu window shows.
3. Click **Export → EAR file**.
4. Save the new itfim.ear to a local file. For example:

   ```
   /tmp/itfim.ear
   ```
5. Backup the original file:

   ```
   /opt/IBM/pkg/release/itfim.ear
   ```

   Complete this backup on the WebSphere Network Deployment manager (when working in a cluster environment) or on the WebSphere Application Server (when using a standalone WebSphere Application Server).
6. Copy the new itfim.ear file to the original location of itfim.ear. For example:

   ```
   # cp /tmp/itfim.ear /opt/IBM/pkg/release/itfim.ear
   ```
7. Continue to "Deploying the updated provisioning runtime component."

Deploying the updated provisioning runtime component

You must deploy the updated provisioning runtime component into WebSphere and configure it into Tivoli Access Manager.

About this task

After you have updated the runtime component, as described in "Adding WS-Security to the provisioning runtime component" on page 184, complete the following steps in this procedure:

Procedure

1. "Unconfiguring and removing the currently deployed runtime"
2. "Deploying and configuring the updated IBM Tivoli Federated Identity Manager runtime component" on page 188

Unconfiguring and removing the currently deployed runtime

Use the unconfigure step to remove the IBM Tivoli Federated Identity Manager runtime from the configuration for Tivoli Access Manager.

About this task

Note: If you have not previously deployed the IBM Tivoli Federated Identity Manager runtime, you can skip this procedure.

The unconfigure step includes the removal of the application identity (for the IBM Tivoli Federated Identity Manager application) from the Tivoli Access Manager domain.

You must remove the configuration for a node before you can remove the runtime application from WebSphere. In a WebSphere cluster environment, you must unconfigure every node before you can remove the runtime.

Procedure

1. Click **Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**.
2. Select the check box in the **Select** column next to the runtime you want to unconfigure. The **Unconfigure** button is now activated.
3. Click **Unconfigure**. The Runtime Node configuration is removed.
4. If you want to remove the runtime, and you have deployed IBM Tivoli Federated Identity Manager into a WebSphere cluster environment, repeat the previous step for every node in the cluster. When the process completes, the Runtime Nodes portlet is redrawn.
5. Verify that the check marks have been removed from the **Configured** and **Enabled** fields for the table entry.

Results

When you have removed the configuration for a runtime, you can remove the runtime component files.

1. Click **Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**. The Current Domain portlet and the Runtime Nodes portlet opens.
2. In the Runtime Nodes portlet, click **Remove Runtime**.

   The runtime is removed from the domain. No further input is required. Verify that the **Remove Runtime** button is deactivated upon completion of the process.

   A message shows that indicates that the runtime has been successfully removed from all the nodes.
3. Continue to "Deploying and configuring the updated IBM Tivoli Federated Identity Manager runtime component."

Deploying and configuring the updated IBM Tivoli Federated Identity Manager runtime component

Use the IBM Tivoli Federated Identity Manager administration console to deploy and configure the updated IBM Tivoli Federated Identity Manager runtime component.

About this task

Attention: If you already have a runtime deployed, you must first remove it. See "Unconfiguring and removing the currently deployed runtime" on page 187.

Procedure

1. Verify that you have copied the modified itfim.ear file into the expected installation location:

   `/opt/IBM/FIM/pkg/release/itfim.ear`

   This step was completed as part of the instructions in "Exporting the IBM Tivoli Federated Identity Manager project" on page 187.
2. Deploy the new runtime file as a WebSphere application. Summary of the configuration steps:
   a. At the IBM Tivoli Federated Identity Manager console, click **Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**. The Runtime Nodes portlet opens.
   b. The Runtime Nodes portlet contains a section titled Runtime Management. This section shows runtime information and contains a button for deploying the runtime. Click **Deploy Runtime**. The runtime is deployed as a WebSphere Application Server application. This might take a few seconds.
   c. Restart the WebSphere server.

      For complete instructions see the topic *Deploying the runtime as a WebSphere Application Server application* in the *Domains* section of the *IBM Tivoli Federated Identity Manager Administration Guide*.
3. Configure the new runtime component into the Tivoli Access Manager environment.

   Summary of instructions:
   a. Click **Tivoli Federated Identity Manager → Domain Management → Runtime Node Management**. The Runtime node table shows the deployed runtimes.
   b. Select the check box in the **Select** column next to the node that you want to configure. The **Configure** button is now activated.
   c. Click **Configure**.

      Note: In a cluster environment, this must be done for *every* node in the table.
      The Runtime Node configuration is started. No further input is required. When the runtime configuration completes, the Runtime Nodes portlet is redrawn.

      For complete instructions see the topic *Configuring the runtime into Tivoli Access Manager*in the *Domains* section of the *IBM Tivoli Federated Identity Manager Administration Guide*.

Chapter 20. Provisioning demonstration scenario

To use the provisioning demonstration scenario, complete the instructions in each of the following topics:

- "Configuring the demonstration scenario"
- "Running the provisioning demonstration scenario" on page 198
- "Verifying provisioning demo" on page 199

Configuring the demonstration scenario

You must deploy several demonstration files and then configure properties files and constants files for the demonstration scenario.

About this task

The deployment steps are dependent on the distributed application topology, and the configuration steps are specific to either the client side or the server side.

Before configuring the demonstration, you must complete configuration of the provisioning service. Verify that you have completed the instructions in Chapter 19, "Configuring provisioning," on page 181.

To configure the demonstration, complete each of the following instructions:

Procedure

1. "Deploying the demonstration file for IBM Tivoli Directory Integrator"
2. "Deploying the demonstration scenario files" on page 192
3. "Configuring the client side" on page 193
4. "Configuring the server side" on page 195

Deploying the demonstration file for IBM Tivoli Directory Integrator

IBM Tivoli Federated Identity Manager provides a demonstration scenario WS-Provisioning `JAR` file for use by IBM Tivoli Directory Integrator.

About this task

The WS-Provisioning JAR file is located in:

```
provisioning/scenario/itfim-provisioning-scenario.jar
```

The file must be copied from the default installation directory to the IBM Tivoli Directory Integrator `jars` directory.

The IBM Tivoli Federated Identity Manager runtime component is always installed on a system that also hosts WebSphere Application Server IBM Tivoli Directory Integrator can be installed on the same (local) system or on a remote system. This means that you must create either a *local* copy or a *remote* copy of the JAR file, depending on your deployment.

You can create a local copy in the following deployment environments:

- When IBM Tivoli Directory Integrator is used in a WebSphere Application Server single-server deployment, and is installed on the *same* system as the WebSphere Application Server.
- When IBM Tivoli Directory Integrator is to be used in a WebSphere Application Server cluster deployment, and is installed on the *same* system as the Deployment Manager.

You must create a remote copy in the following deployment environments:

- When IBM Tivoli Directory Integrator is used in a WebSphere Application Server single-server deployment, and is installed on a *different* system as the WebSphere Application Server.
- When IBM Tivoli Directory Integrator is used in a WebSphere Application Server cluster deployment, and is installed on a *different* system than the Deployment Manager.

  Note: The IBM Tivoli Directory Integrator can be located either on one of the cluster nodes, or on a host system that is not configured into the cluster.

Procedure

1. Complete the file copy:
   - UNIX

     Copy 2 files:

     ```
     /opt/IBM/FIM/provisioning/scenario/itfim-provisioning-scenario.jar
     /opt/IBM/FIM/provisioning/idijars/itfim-provisioning.jar
     ```

     into the appropriate directory: (for):

     – Tivoli Directory Integrator 6.0

       ```
       /opt/IBM/IBMDirectoryIntegrator/jars
       ```

     – Tivoli Directory Integrator 6.1.1

       ```
       /opt/IBM/TDI/V6.1.1/jars
       ```

   - Windows

     Copy:

     ```
     C:\Program Files\opt\IBM\FIM\provisioning\scenario\
        itfim-provisioning-scenario.jar
     C:\Program Files\opt\IBM\FIM\provisioning\idijars\itfim-provisioning.jar
     ```

     into the appropriate directory: (for):

     – Tivoli Directory Integrator 6.0

       ```
       C:\Program Files\opt\IBM\IBMDirectoryIntegrator\jars
       ```

     – Tivoli Directory Integrator 6.1.1

       ```
       C:\Program Files\opt\IBM\TDI\V6.1.1\jars
       ```

2. Continue with "Deploying the demonstration scenario files."

Deploying the demonstration scenario files

When you want to use the provisioning demonstration scenario in a WebSphere Application Server *cluster* environment, you must manually deploy the demonstration scenario files.

About this task

Note: If you are deploying the provisioning demonstration scenario into a WebSphere Application Server *single-server* environment, you can skip this topic. Continue with "Configuring the client side" or "Configuring the server side" on page 195, as appropriate.

The provisioning demonstration scenario files are by default installed in:
`provisioning/scenario/*`

The provisioning demonstration scenario files are installed as part of the IBM Tivoli Federated Identity Manager runtime component. When the runtime component is installed into a WebSphere Application Server cluster environment, the runtime component files are programmatically deployed (pushed out) to each node in the cluster.

However, the provisioning demonstration scenario files are *not* programmatically deployed. You must manually copy the files to each node in the cluster.

Procedure

1. For each node:
   a. Create the subdirectory `scenario` within the `provisioning` directory.
      The `provisioning` directory exists before you copy `itfim-provisioning.ear` to the node.
   b. Copy all of the scenario files from the Deployment Manager to the `provisioning/scenario` directory on the node.
2. Continue with either "Configuring the client side" or "Configuring the server side" on page 195, depending on the role of the demonstration scenario provisioning service.

Configuring the client side

The steps in configuring the demonstration scenario is specific to its role as a client.

Procedure

1. Configure the IBM Tivoli Directory Server change log.
   IBM Tivoli Directory Server supports creation of a change log database. This database is used to record changes to the schema or directory entries in a typical LDAP entry structure. The change log records all update operations, such as add, delete, modify, and modrdn.
   IBM Tivoli Directory Integrator operates as a client application of IBM Tivoli Directory Server, and uses the change log to retrieve a set of changes that have been made to the IBM Tivoli Directory Server database. The activities logged in the change log are used by IBM Tivoli Directory Integrator to trigger provisioning events.
   You can use either graphical Configuration tool or the **`idscfgchglg`** command-line utility to enable the change log.

   Note: The preceding instructions apply to IBM Tivoli Directory Server 6.0. When you are using IBM Tivoli Directory Server 5.2, the commands are different.
   The Configuration tool is described in the *IBM Tivoli Directory Server Installation Guide*. The command-line utility is described in both the *IBM Tivoli Directory*

Server Installation Guide and the *IBM Tivoli Directory Server Administration Guide*. See the product documentation for your version of IBM Tivoli Directory Server.

Note: You can access further instructions at the IBM Tivoli Directory Server Information Center: http://publib.boulder.ibm.com/infocenter/tiv2help/index.jsp?toc=/com.ibm.IBMDS.doc/toc.xml

2. Update the settings in the client-side properties file for the example scenario:
   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/ITFIMClientIDI.properties`
   - Windows

     `C:\Program Files\opt\IBM\FIM\provisioning\scenario\ITFIMClientIDI.properties`

   Provide a value for each of the following entries:

   ldapURL
   The URL for the client LDAP server. For example:

   ```
   ldapURL:ldap://idp.example.com:389
   ```

   ldapAdmin
   The LDAP administrative user for the client LDAP server. For example:

   ```
   ldapAdmin:cn=root
   ```

   ldapAdminPwd
   The password for the LDAP administrative user for the client LDAP server. For example:

   ```
   ldapAdminPwd:passw0rd
   ```

   searchBase
   The LDAP suffix that forms the base of the hierarchy to be searched. For example:

   ```
   searchBase:c=us
   ```

   providerURL
   The URL for the target of the provisioning request. In most cases, this is the client-side WebSphere Application Server provisioning application. For example:

   ```
   providerURL:http://idp.example.com:9080/wsp/wspservice
   ```

   In a WebSphere cluster environment, this would be the HTTP server with the WebSphere Application Server plug-in that routes requests to the correct node server. For example:

   ```
   providerURL:http://idp-apache.example.com/wsp/wspservice
   ```

   Figure 15 on page 195 shows the additional entries in the client properties file that typically do not need to be changed for the example scenario.

```
# LDAP search parameter
searchScope:subtree

# Changelog base
changeLogBase:cn=changelog

# Initial change number
initialChangeNumber:0

# Authentication method
authenticationMethod:Anonymous

# The location of the wsp.wsdl. Relative to the working directory
wsdlURL:itfim_wsp_wsdl/wsp.wsdl

# The location of the LDAP change log filename. Keeps an record of LDAP updates
changeNumberFilename:changelog_number.txt
```

Figure 15. Client properties that typically do not need to change for the demonstration scenario

3. (Optional) Update the setting in the constants file for the example scenario. This setting is required to run the IBM Tivoli Federated Identity Manager scenario on the client-side server.

   Location of the constants file:

   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/ITFIMClientScript.constants`

   - Windows

     `C:\Program Files\opt\IBM\FIM\provisioning\scenario\ITFIMClientScript.constants`

   The file contains the name of the target company:

   `fim.wsprovisioning.target=BenefitsCompany TAM`

Configuring the server side

The steps in configuring the demonstration scenario is specific to its role as a server.

Procedure

1. Update the settings in the server-side properties file for the example scenario:

   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/ITFIMServerIDI.properties`

   - Windows

     `C:\Program Files\opt\IBM\FIM\provisioning\scenario\ITFIMServerIDI.properties`

   Provide a value for each of the following entries:

   serverPort
   The port on which IBM Tivoli Directory Integrator listens for SOAP requests. The example configuration file uses port 9080. Change port to another value, since port 9080 is typically used by WebSphere. For example:

   `serverPort:8888`

   wsdlURL
   The location of the WS-Provisioning WSDL file. The location is relative to the working directory. Do not change this value. Default:

   `wsdlURL:itfim_wsp_wsdl/wsp.wsdl`

2. Update the settings in the server constants file for the example scenario:
   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/ITFIMServerScript.constants`
   - Windows

     `C:\Program Files\opt\IBM\FIM\provisioning\scenario\ITFIMServerScript.constants`

   You must provide a value for each of the following entries:

   fim.wsprovisioning.target.ldap.prefix
   The prefix for the target LDAP. For example:

   `fim.wsprovisioning.target.ldap.prefix=cn=`

   fim.wsprovisioning.target.ldap.suffix
   The suffix for the target LDAP server. For example:

   `fim.wsprovisioning.target.ldap.suffix=o=ibm,c=au`

   An LDAP suffix that you must create on the LDAP server. This suffix is the repository for all provisioning-related users. When a new user is created through WS-Provisioning, the Distinguished Name (DN) of the user on the server side is created by combining the following values:
   - The value of `fim.wsprovisioning.target.ldap.prefix`. For example:

     `cn=`
   - The user name. For example:

     `johndoe`
   - The value of `fim.wsprovisioning.target.ldap.suffix`. For example:

     `o=ibm,c=au`

   In this example, the DN is:

   `cn=johndoe,o=ibm,c=au`

   fim.wsprovisioning.tam.config.url
   The URL for the generated configuration file generated by the Tivoli Access Manager **SvrSslCfg** utility. For example:
   - UNIX

     ```
     fim.wsprovisioning.tam.config.url=file:///opt/IBM/FIM/provisioning/
        scenario/amconfig.properties
     ```
   - Windows

     ```
     fim.wsprovisioning.tam.config.url=file:///c:\\Progra~1\\IBM\\FIM\\
        provisioning\\scenario\\amconfig.properties
     ```

   fim.wsprovisioning.tam.admin
   The Tivoli Access Manager administrative user. The default user is sec_master. For example:

   `fim.wsprovisioning.tam.admin=sec_master`

   fim.wsprovisioning.tam.admin.pwd
   The password for the Tivoli Access Manager administrative user. For example:

   `fim.wsprovisioning.tam.admin.pwd=passw0rd`

   fim.wsprovisioning.target.ldap
   The URL for the target LDAP server. For example:

   `fim.wsprovisioning.target.ldap=ldap://localhost`

   fim.wsprovisioning.target.ldap.admin
   The administrative user for the LDAP server. For example:

   `fim.wsprovisioning.target.ldap.admin=cn=root`

fim.wsprovisioning.target.ldap.admin.pwd
The password for the administrative user for the LDAP server. For example:

```
fim.wsprovisioning.target.ldap.admin.pwd=passw0rd
```

Figure 16 shows additional entries in the server constants file that typically do not need to be changed for the example scenario.

```
# The password for any generated TAM users
fim.wsprovisioning.provisioned.user.pwd=passw0rd

# The target company name
fim.wsprovisioning.target=BenefitsCompany TAM

# The LDAP unique identifier field
fim.wsprovisioning.target.ldap.uid=uniqueIdentifier
```

Figure 16. Server constants values that do not need to change

3. The IBM Tivoli Federated Identity Manager provisioning scenario provides a configuration script that configures the Java Runtime Environment of the IBM Tivoli Directory Integrator for use with Tivoli Access Manager. You must edit the configuration script to specify values for your deployment:
   - UNIX

     ```
     /opt/IBM/FIM/provisioning/scenario/config_idi_jre.sh
     ```
   - Windows

     ```
     C:\Program Files\opt\FIM\provisioning\scenario\config_idi_jre.bat
     ```

   Specify the following values:

   TAM_ADMIN_ID
   The Tivoli Access Manager administrative user. For example:

   ```
   TAM_ADMIN_ID=sec_master
   ```

   TAM_ADMIN_PWD
   The password for the Tivoli Access Manager administrative user. For example:

   ```
   TAM_ADMIN_PWD=passw0rd
   ```

   POLICYSVR
   The host name and port for the Tivoli Access Manager policy server. For example:

   ```
   POLICYSVR=localhost:7135:1
   ```

   AUTHSVR
   The host name and port for the Tivoli Access Manager authorization server. For example:

   ```
   AUTHSVR=localhost:7136:1
   ```

   Figure 17 on page 198 shows the contents of the configuration script. The script completes the configuration of the provisioning application into Tivoli Access Manager.

```
TAM_ADMIN_ID=sec_master
TAM_ADMIN_PWD=passw0rd

POLICYSVR=localhost:7135:1
AUTHSVR=localhost:7136:1

# First we run the PDJrteCfg to configure the JRE for use with TAM. This can
# also be done manually with the pdconfig utility.

echo "Configuring IDI JRE for use with TAM"

/opt/PolicyDirector/sbin/pdjrtecfg -action config -host localhost -port 7135
     -java_home /opt/IBM/IBMDirectoryIntegrator/_jvm/jre -config_type full

# Now we run the SvrSslCfg utility to create an identity for the IDI Server
# within the TAM registry.

echo "Configuring a TAM server identity for IDI"

/opt/IBM/IBMDirectoryIntegrator/_jvm/jre/bin/java com.tivoli.pd.jcfg.SvrSslCfg
-action config -admin_id $TAM_ADMIN_ID -admin_pwd $TAM_ADMIN_PWD
-appsvr_id itfimprovisioning -port 999
-mode remote -policysvr $POLICYSVR -authzsvr $AUTHSVR
-cfg_file /opt/IBM/FIM/provisioning/scenario/amconfig.properties
-key_file /opt/IBM/FIM/provisioning/scenario/amkey.jks
```

Figure 17. Server-side configuration script for Tivoli Access Manager Java runtime environment

4. Run the configuration script.

Running the provisioning demonstration scenario

You must run scripts separately to execute the server and client-side scenarios.

About this task

Complete the following instructions:

Procedure

1. On the client side computer, run the script that executes the client-side scenario:
   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/runclient.sh`
   - Windows

     `C:\Program Files\opt\FIM\provisioning\scenario\runclient.bat`

   Figure 18 shows the contents of the UNIX version of the runclient script (runclient.sh):

```
#!/bin/sh
/opt/IBM/IBMDirectoryIntegrator/ibmdisrv -s "/opt/IBM/FIM/provisioning/scenario"
  -c "ITFIMClient.xml" -t fim_ldap_handler
```

Figure 18. Contents of the runclient script

2. On the server side computer, run the script that executes the server side scenario:
   - UNIX

     `/opt/IBM/FIM/provisioning/scenario/runserver.sh`

- Windows

  ```
  C:\Program Files\opt\FIM\provisioning\scenario\runserver.bat
  ```

Figure 19 shows the contents of the UNIX version of the runserver script (runserver.sh):

```
#!/bin/sh
/opt/IBM/IBMDirectoryIntegrator/ibmdisrv -s "/opt/IBM/FIM/provisioning/scenario"
   -c "ITFIMServer.xml" -r "FIM_WS-Provisioning Server"
```

Figure 19. Contents of the runserver script

Verifying provisioning demo

This topic outlines three tasks on how you can verify the provisioning demo.

- "Verifying provisioning demo user create"
- "Modifying a demonstration user"
- "Verifying provisioning demonstration user delete" on page 200

Verifying provisioning demo user create

This topic discusses how you can verify if the user creation was successfully provisioned.

Procedure

1. On the client side system, log in to **pdadmin**.
2. Create a test user. For example:

   ```
   pdadmin> user create testuser cn=testuser,c=us passw0rd passw0rd passw0rd
   ```

   The creation of the user also occurs on the LDAP server. This user creation triggers a change in the change log number. This change is detected by the IBM Tivoli Directory Integrator event handler on the listening application of the client.

   The client side IBM Tivoli Directory Integrator contacts the server-side IBM Tivoli Directory Integrator with a request to create the same user. The server-side IBM Tivoli Directory Integrator example customer component uses the Tivoli Access Manager Java administration API to provision a Tivoli Access Manager user into the server-side LDAP.
3. To verify that the user was successfully provisioned:
   a. Log on to **pdadmin** on the server-side Tivoli Access Manager system.
   b. Use **pdadmin** to show the information about the user you just created. For example.

      ```
      pdadmin> user show testuser
      ```

Modifying a demonstration user

This topic discusses how you can verify if the user modification was successfully provisioned.

Procedure

1. On the client side system, log on to **pdadmin**.
2. Set the testuser account that you created in "Verifying provisioning demo user create" to the valid state.

   ```
   pdadmin> user modify testuser account-valid yes
   ```
3. Verify that the same user on the server side also gets modified.

Verifying provisioning demonstration user delete

This topic discusses how you can verify if the user deletion was successfully provisioned.

Procedure

1. On the client side system, log on to **pdadmin**.
2. Delete the user that you created. For example:

   ```
   pdadmin> user delete -registry testuser
   ```

3. To verify that the user was successfully deleted:
   a. Log on to **pdadmin** on the server-side Tivoli Access Manager system.
   b. Use **pdadmin** to display information about the user you just deleted. For example.

      ```
      pdadmin> user show testuser
      ```

Chapter 21. Command reference

You can use IBM Tivoli Federated Identity Manager commands, instead of the console, to perform many configuration and administrator tasks.

The commands are integrated into the WebSphere Application Server Command Manager Framework. For more information about this command framework, see the WebSphere Application Server 8.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.

Use either of the following commands to run the IBM Tivoli Federated Identity Manager:

- The WebSphere Application Server wsadmin tool using a command line. Type the appropriate command for your operating system to start the tool:

  Windows
  `wsadmin.bat`

  AIX, Linux, or Solaris
  `wsadmin.sh`

  Note: For more information about the options that can be specified when you run the wsadmin tool, see the WebSphere Application Server 8.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.
- A Java Management Extension (JMX) application.

  Note: The use of JMX requires development of the appropriate programming applications. For more information, see the *Administrative command framework programming guide for WebSphere Application Server* at http://www.ibm.com/developerworks/websphere/techjournal/0610_chang/0610_chang.html.

Attention: If you have installed IBM Tivoli Federated Identity Manager, ensure that you stop WebSphere Application Server and then restart it before attempting to run any of the IBM Tivoli Federated Identity Manager commands.

Attention: If the **TFIMCommands** plug-in does not start automatically, perform the following steps:

1. Stop the WebSphere Application Server.
2. From the console, issue the following command: `/opt/IBM/WebSphere/AppServer/profiles/profile_name/bin/osgiCfgInit.sh`
3. Restart the WebSphere Application Server.
4. Open the WebSphere Application Server wsadmin tool.
5. In wsadmin, issue the following command: `$AdminTask help -commandGroups`
6. Confirm that **TFIMCommands** is present in the list of **commandGroups**.

Location of commands

The IBM Tivoli Federated Identity Manager commands are packaged in a single .jar file named `com.tivoli.am.fim.commands.jar`.

The location of the `.jar` file depends on the WebSphere Application Server version you use in installing the runtime and management services component:

WebSphere Application Server 6.1, WebSphere Application Server 7.0, and WebSphere Application Server 8.0, including the embedded version
The plug-ins directory, such as `/opt/IBM/WebSphere/AppServer/plugins`

WebSphere Application Server 6.0
The lib directory, such as `/opt/IBM/WebSphere/AppServer/lib`

Listing available commands

To list the available IBM Tivoli Federated Identity Manager commands, start the wsadmin tool and then type the following command and press Enter:

```
$AdminTask help TFIMCommands
```

Getting help for a specific command

To get help for a specific IBM Tivoli Federated Identity Manager command, start the wsadmin tool and then type the following command and press Enter:

```
$AdminTask help command_name
```

For example, to see the help for the managing domains command, **manageItfimDomain**, type: `$AdminTask help manageItfimDomain`

Requesting parameter prompts for a specific command

To be prompted for the required and optional parameters as you run a command, start the wsadmin tool and then type the following command and press **Enter**:

```
$AdminTask command_name -interactive
```

For example, to show prompts when you run the **manageItfimDomain**, type:

```
$AdminTask manageItfimDomain -interactive
```

Tasks supported by the commands

IBM Tivoli Federated Identity Manager commands are available for the following tasks:

Managing domains

Command name: **manageItfimDomain**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on a domain:

- list
- view
- create
- delete
- import
- export
- deploy
- undeploy
- configure

- unconfigure
- publish plug-ins
- publish pages

Managing federations

Command name: **manageItfimFederation**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on a federation:

- list
- create (using a response file)
- create a response file
- remove
- view
- export
- modify

Managing partners

Command name: **manageItfimPartner**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on a partner:

- list
- create (using a response file)
- create a response file
- remove
- view
- enable
- disable
- export
- modify

Managing point of contact servers

Command name: **manageItfimPointOfContact**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on a point of contact:

- list
- listCallbacks
- create (using a response file)
- create a response file
- view
- activate
- delete

Managing keys and keystores

Command name: **manageItfimKeys**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on the keys and keystores in your environment:

- list
- delete
- enable
- disable
- import
- export

Managing the alias service

Command name: **manageItfimNameIdSvc**

Purpose: This command, when used with the appropriate parameters, can perform the following operations on the alias service in your environment:

- view
- configure
- add a host
- remove a host
- modify a host

Managing reports

Command name: **manageItfimReports**

Purpose: This command, when used with the appropriate parameters, can perform the following operations for viewing, running, and deleting reports:

- listActive
- listArchived
- listRunnable
- create the response file
- delete
- run

Reloading the management service

Command name: **reloadIfimManagementService**

Purpose: This command reloads the IBM Tivoli Federated Identity Manager management service.

Reloading the runtime

Command name: **reloadIfimRuntime**

Purpose: This command reloads the IBM Tivoli Federated Identity Manager runtime.

Logging out a SAML 2.0 user

Command name: **logoutSaml20User**

Purpose: This command logs out the user of a SAML 2.0 single sign-on session.

Defederating a SAML 2.0 user:

Command name: **defederatetSaml20User**

Purpose: This command defederates a user from a SAML 2.0 federation.

Managing the artifact resolution service

Command name: **manageItfimSamlArtifactService**

Purpose: This command, when used with the appropriate parameters, can perform the following operations to manage the SAML 1.x Artifact Service:

- list
- configure
- unconfigure

Managing an STS module type

Command name: **manageItfimStsModuleType**

Purpose: This command, when used with the appropriate parameters, can perform the following operations to manage token module types:

- list
- view

Managing an STS module instance

Command name: **manageItfimStsModuleInstance**

Purpose: When used with the appropriate parameters, this command can perform the following operations to manage token module instances:

- list
- view
- delete
- create (using a response file)
- create the response file
- modify

Managing an STS chain mapping

Command name: **manageItfimStsChainMapping**

Purpose: This command manages the chain mapping identification associated with a trust chain. When used with the appropriate parameters, this command can perform the following operations to manage STS chain mapping:

- list
- view
- delete
- create (using a response file)
- create the response file
- modify

Managing an STS chain

Command name: **manageItfimStsChain**

Purpose: This command, when used with the appropriate parameters, can perform the following operations to manage trust service chains:

- list
- view
- delete
- create (using a response file)
- create the response file

Administration commands that replace the staging tools

The command-line tools described in this command reference replace the IBM Tivoli Federated Identity Manager staging utilities.

The previous IBM Tivoli Federated Identity Manager versions contained a set of scripts known as the staging utilities. These scripts provided part of the function of a command-line interface, and were used for two purposes:

- To replicate servers, as a command-line alternative to the import and export functions in the administration console.
- To migrate configuration information from one IBM Tivoli Federated Identity Manager source domain to a target domain.

IBM Tivoli Federated Identity Manager now contains a full-featured command-line interface that can be used to accomplish these tasks. The command-line interface is more robust and supports more options.

The command-line interface supports:

- Response files that replace the protocol-specific properties files used by the staging tools.
- The **manageItfimDomain** command, which provides a superset of the domain management functions included in the staging tools, such as domain creation, import, export, and configure.
- The **manageItfimFederation** command, which supports creation, deletion, modification, and export of federations.
- The **manageItfimPartner** command, which supports creation, deletion, modification, and export of federation partners.

manageItfimDomain

Use the **manageItfimDomain** command to manage domains.

Purpose

The **manageItfimDomain** command can perform the following operations on a domain when used with the appropriate parameters:

- list
- view
- create
- delete
- import
- export
- deploy
- undeploy
- configure
- unconfigure
- publish plug-ins
- publish pages

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimDomain {-operation operator [optional_parameters]}
```

where the `-operation` parameter and its value *operator* are required. The optional parameters are:

```
-fimDomainName name

-clusterId cluster
-fileId file
-tamAdminId ID
-tamAdminPwd password
-tamPolicyServer hostname
-tamPolicyPort port
-tamAuthzServers hostname1,hostname2...
-tamAuthzPorts port1,port2...
-tamDomainName name
```

Parameters

The following parameters are available for use with the **manageItfimDomain** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform on the domain. Valid values are described in the following table.

Table 18. Values for the manageItfimDomain -operation parameter

Value	Description and requirements
list	List all of the existing domains.
view	View the details of a domain. When you use this operator, you must also use the **fimDomainName** parameter.

Table 18. Values for the manageItfimDomain -operation parameter (continued)

Value	Description and requirements
create	Create a new domain. When you use this operator, you must also use the following parameters: **fimDomainName** *name* **clusterId** *cluster* If the domain uses Tivoli Access Manager, the following parameters are also required: **tamAdminId** *ID* **tamPolicyServer** *hostname* **tamPolicyPort** *port* **tamAuthzServers** *hostname1,hostname2...* **tamAuthzPorts** *port1,port2...* **tamDomainName** *name* **Note:** If you create a domain using this command and later you want to work with the domain using the console in a browser, you must make the console program aware of the domain by clicking **Domains** > **Create** in the console. The domain you created using the command should then be listed in the console.
delete	Remove a domain. When you use this operator, you must also use the When you use this operator, you must also use the following parameters: **fimDomainName** *name* **clusterId** *cluster*
import	Import a domain. When you use this operator, you must also use the **fimDomainName** and **fileId** parameters.
export	Export a domain. When you use this operator, you must also use the **fimDomainName** and **fileId** parameters.
importRuntimeCustomProperties	Import runtime custom properties. When you use this operator, you must also use the **fimDomainName** and **fileId** parameters.
exportRuntimeCustomProperties	Export runtime custom properties. When you use this operator, you must also use the **fimDomainName** and **fileId** parameters.
deploy	Deploy a domain. When you use this operator, you must also use the **fimDomainName** parameter.
undeploy	Undeploy a domain. When you use this operator, you must also use the **fimDomainName** parameter.

Table 18. Values for the manageItfimDomain -operation parameter (continued)

Value	Description and requirements
Note: The deploy and undeploy operations can take a long time to complete. The wsadmin SOAP connection might time out before the operation is finished. Even if this timeout occurs in the wsadmin client, the operation will complete on the management service. To avoid timeouts, you can increase the SOAP request timeout, as follows: 1. Modify the com.ibm.SOAP.request.Timeout property to 800. The property is in the WebSphere installation directory and the following subdirectory: /profiles/*profile_name*/ properties /soap.client.props 2. Restart the WebSphere server.	
configure	Configure a domain. When you use this operator, you must also use the following parameters: **fimDomainName** *name* If the domain that you are configuring uses Tivoli Access Manager, the following parameter is required: **tamAdminPwd** *password*
unconfigure	Unconfigure a domain. When you use this operator, you must also use the following parameters: **fimDomainName** *name* If the domain that you are unconfiguring uses Tivoli Access Manager, the following parameter is required: **tamAdminPwd** *password*
publishPlugins	Publish the plug-ins for custom point of contact server modules or custom identity mapping modules. When you use this operator, you must also use the **fimDomainName** parameter.
publishPages	Publish custom event pages or page locales. When you use this operator, you must also use the **fimDomainName** parameter.

-fimDomainName *name*
This parameter is required for all operations except **list**. The value used with this parameter is the name of the domain on which the operation will be performed. The name must be a string with characters of any type.

-clusterId *cluster*
This parameter is required when you create or delete a domain. The value used with this parameter is the cluster or server name where the domain is located or will be created. The name must be a string and must be in the either of the following forms:

For a server:

WebSphere:cell=<cell_name>,node=<node_name>,server=<server_name>

For a cluster:

WebSphere:cell=<cell_name>, cluster=<cluster_name>

-fileId *output_file* | *input_file*
This parameter is required when you import or export a domain. The value used with this parameter is the file name and path to which the domain will be exported (output file) or the file that is being imported (input file). The path and file name must be valid for the operating system being used.

-tamAdminId *ID*
This parameter is required when you create a domain in which Tivoli Access Manager is used. The value used with this parameter is the ID of the Tivoli Access Manager administrator. The ID must be a string with characters of any type.

-tamAdminPwd *password*
This parameter is required when you configure or unconfigure a domain in which Tivoli Access Manager is used. The value used with this parameter is the password of the Tivoli Access Manager administrator. The password must be a string with characters of any type.

-tamPolicyServer *hostname*
This parameter is required when you create a domain in which Tivoli Access Manager is used. The value used with this parameter is the hostname or IP address of the Tivoli Access Manager policy server. The hostname must be a string with characters of any type.

-tamPolicyPort *port*
This parameter is required when you create a domain in which Tivoli Access Manager is used. The value used with this parameter is the port number where the Tivoli Access Manager policy server is listening. The port must be numeric unsigned integer values.

-tamAuthzServers *hostname1,hostname2,*
This parameter is required when you create a domain in which Tivoli Access Manager is used. The value used with this parameter is a list of one or more Tivoli Access Manager authorization servers (the hostnames or the IP addresses). Each hostname or IP address must be a string with characters of any type. Separate multiple values with commas.

-tamAuthzPorts *port1,port2,*
This parameter is required when you create a domain. The value used with this parameter is a list of one or more port numbers on which the Tivoli Access Manager authorization servers are listening. Each port must be numeric unsigned integer values. Separate multiple values with commas.

-tamDomainName *name*
This parameter is required when you create a domain in which Tivoli Access Manager is used. The value used with this parameter is the name of the Tivoli Access Manager domain. The name must be a string with characters of any type. The default value is `Default`.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List the existing domains:

```
$AdminTask manageItfimDomain {-operation list}
```

View the metadata of a domain:

```
$AdminTask manageItfimDomain {-operation view -fimDomainName domain1}
```

Create a domain:

```
$AdminTask manageItfimDomain {-operation create -fimDomainName domain1
  -clusterId WebSphere:cell=cell01,node=node01,server=server1
  -tamAdminId sec_master
  -tamPolicyServer tam1.example.com -tamPolicyPort 7135
  -tamAuthzServers tam2.example.com,tam3.example.com
  -tamAuthzPorts 7136, 7137 -tamDomainName Default}
```

Delete a domain:

```
$AdminTask manageItfimDomain {-operation delete -fimDomainName domain1
  -clusterId WebSphere:cell=cell02,cluster=cluster02}
```

Import a domain:

```
$AdminTask manageItfimDomain {-operation import fimDomainName domain1
  -fileId c:\temp\domain1.rsp}
```

Export a domain:

```
$AdminTask manageItfimDomain {-operation export -fimDomainName domain1
  -fileId c:\temp\domain1.rsp}
```

Import runtime custom properties:

```
$AdminTask manageItfimDomain {-operation importRuntimeCustomProperties
-fimDomainName domain1 -fileId /tmp/runtimeProperties.xml}
```

Export runtime custom properties:

```
$AdminTask manageItfimDomain {-operation exportRuntimeCustomProperties
-fimDomainName domain1 -fileId /tmp/runtimeProperties.xml}
```

Deploy the runtime into a domain:

```
$AdminTask manageItfimDomain {-operation deploy -fimDomainName domain1}
```

Undeploy the runtime from a domain:

```
$AdminTask manageItfimDomain {-operation undeploy -fimDomainName domain1}
```

Configure a domain

```
$AdminTask manageItfimDomain {-operation configure -fimDomainName domain1
  -tamAdminPwd passw0rd}
```

Unconfigure a domain:

```
$AdminTask manageItfimDomain {-operation unconfigure -fimDomainName domain1
  -tamAdminPwd passw0rd}
```

Publish plug-ins:

```
$AdminTask manageItfimDomain {-operation publishPlugins
  -fimDomainName domain1}
```

Publish pages:

```
$AdminTask manageItfimDomain {-operation publishPages
  -fimDomainName domain1}
```

manageItfimFederation

Use the **manageItfimFederation** command to manage federations.

Purpose

The **manageItfimFederation** command, when used with the appropriate parameters, can perform the following operations on a federation:

- list
- create (using a response file)
- create the response file
- delete
- view

- modify
- export

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimFederation {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-federationName name
-protocol type
-federationRole role
-mapModuleInstanceId ID
-fileId output_file | input_file
```

The use of these parameters depends on the operator you choose.

Parameters

The following parameters are available for use with the **manageItfimFederation** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 19. Values for the manageItfimFederation -operation parameter

Value	Description and requirements
list	Lists all of the existing federations.

Table 19. Values for the manageItfimFederation -operation parameter (continued)

Value	Description and requirements
createResponseFile **Note:** If you are creating a new federation that is not based on an existing federation, you might find it more expedient to create the federation using the console, as described in the Configuration Guide. Then to make minor modifications, you could run the command for creating a response file based on that existing federation, and edit the response file that results from that command, before you run the command for creating the federation.	Create a response file to use for creating a federation. You can create a response file for a new federation or create a response file that is based on an existing federation. **New federation:**When you use this operator to create response file for a new federation, you must also specify the following parameters: **protocol** *type* Specify the federation type if you are creating a response file that is not based on an existing federation. **role** *role* Specify the role of the federation if you are creating a response file that is not based on an existing federation. This parameter indicates whether you are the identity provider or the service provider in the federation. **mapModuleInstanceID***ID* Specify the ID of the custom mapping module you want to use. This parameter is optional and is used only if you are not using an XSLT file for identity mapping. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. **Based on existing federation:**When you use this operator to create a response file that is based on an existing federation, you must also specify the following parameters: **federationName** *name* Specify the federation name if you are creating a response file that is based on an existing federation. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. After you have created the response file, open it with a text editor, review the attributes that are defined in the file, make changes as required by your environment, and then save and close the file. For information about the content of the response file, see: • "SAML federation response file reference" on page 216 • "OAuth 1.0 federation response file reference" on page 238 • "OAuth 2.0 federation response file reference" on page 242 • "WS-Federation federation response file" on page 226

Table 19. Values for the manageItfimFederation -operation parameter (continued)

Value	Description and requirements
create	Create a federation using a response file. When you use this operator, you must also specify the following parameters: **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.
delete	Delete a federation. When you use this operator, you must also specify the **federationName** ***name***.
view	View the details of a federation. When you use this operator, you must also specify the **federationName** ***name***.
modify	Modify the properties of a federation. The use of this operation is a three-step process. You must first run the createResponseFile operator and specify an existing federation so that a file containing the properties in that federation are created. Next, you open the response file with a text editor to modify the properties you want to change. Save and close the file. Then, when you use the modify operator, you must specify the **fileId** ***name*** of the edited response file.
export	Export the metadata of a federation. When you use this operator, you must also specify the following parameters: **federationName** *name* The name of the federation you are exporting. **fileId** *input_file* Specify the file name and path for the file that is created for the exported federation.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name can be a string with characters of any type.

-federationName *name*
This parameter is required if you are creating a response file that is based on an existing federation, deleting a federation, viewing a federation, enabling a federation, disabling a federation, or exporting a federation. The value used with this parameter is the name of the federation on which the operation will be performed. The name can be a string with characters of any type.

-protocol *type*
This parameter is required if you are creating a response file that is not based on an existing federation. The value used with this parameter is the type of the federation. The type can be

- SAML2_0
- SAML1_0
- SAML1_1
- OPENID
- OAUTH1_0

- OAUTH2_0
- WSFED

Note: Type value is case sensitive.

-role *role*
This parameter is required if you are creating a response file that is not based on an existing federation. The value used with this parameter is the role of the federation. The role can be either ip or sp.

-mapModuleInstanceId *ID*
This parameter is required if you are creating a federation that will use a custom mapping module instead of an XSLT file for mapping. The ID that you specify is specific to the custom module you want to use.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file, creating a federation, or exporting a federation. The value used with this parameter is the file name and path to which the federation is exported (output file) or a response file will be read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the federations in a domain:

```
$AdminTask manageItfimFederation {-operation list -fimDomainName domain1}
```

Create an empty response file for a new federation that is *not* based on an existing federation:

```
$AdminTask manageItfimFederation {-operation createResponseFile
  -fimDomainName domain1 -protocol SAML2_0 -role ip
  -mapModuleInstanceId default_tdi
  -fileId c:\temp\saml2idp.rsp}
```

The file specified here is the name of the response file that you are creating with the command. You will use this file as input for creating a federation or modifying a federation. After you have created this file, open it with a text editor and define the attributes in it so that they are correct for your environment.

Also, the mapModuleInstanceId is used because in this example the federation will use a custom mapping module instead of an XSLT file.

Note: If you are creating a new federation that is not based on an existing federation, you might find it more expedient to create the federation using the console, as described in the *IBM Tivoli Federated Identity Manager Configuration Guide*. To make minor modifications, you can run the command for creating a response file based on that existing federation, and edit the response file that results from that command, before you run the command for creating the federation.

Create a response file based on an existing federation:

```
$AdminTask manageItfimFederation {-operation createResponseFile
  -fimDomainName domain1 -federationName idpsaml2
  -fileId c:\temp\saml2idp.rsp
}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a federation. After you have created this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a federation:

```
$AdminTask manageItfimFederation {-operation create -fimDomainName domain1
  -fileId c:\temp\saml2idp.rsp}
```

Note: The file specified here is the response file and is used as input. Before running this command, you must have opened the response file with a text editor and ensured that the attributes that are defined in the file are correct for your environment.

Delete a federation:

```
$AdminTask manageItfimFederation {-operation delete -fimDomainName domain1
  -federationName fed1}
```

View federation details:

```
manageItfimFederation {-operation view -fimDomainName domain1
  -federationName fed1}
```

Modify a federation

First run createResponseFile, as described in *Create a response file based on an existing federation*, to create a file that contains all of the properties in the federation. Edit it using a text editor. Save it Then, reload it into your environment using the modify command.

```
$AdminTask manageItfimFederation {-operation modify -fimDomainName domain1
  -fileId c:\temp\saml2idp.xml }
```

Export a federation

```
$AdminTask manageItfimFederation {-operation export -fimDomainName domain1
  -federationName fed1 -fileId c:\temp\saml2idp.xml}
```

SAML federation response file reference

Before you can create a federation using the **manageItfimFederation** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

You can create a response file for a new federation or one that is based on an existing federation.

New federation

Create a response file for creating a new federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -protocol type -role role -fileId output_file}
```

Existing federation

Create a response file for creating a federation that is based on an existing federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the protocol and federation role, either

`identityProvider` or `serviceProvider`, that you specify in the command or that are specified by the properties of the existing federation input file.

The list that follows describes the parameters used in the response files.

Note: If you created a response file that is based on an existing federation, values are automatically specified for many of the parameters.

The following process describes how to edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

Examples of the response files are in the following directories:

AIX, Linux or Solaris

`/opt/IBM/FIM/examples/responsefiles`

Windows

`C:\Program Files\IBM\FIM\examples\responsefiles`

Parameters

Table 20. Parameters in SAML federation response files

Parameter	Value	Description
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
AllowIBMProtocolExtension	`true` or `false`	Setting that specifies whether the use of the IBM Protocol Extension is allowed or not. The extension allows a query-string parameter that specifies whether to use browser artifact or browser POST in a SAML 1.x federation.
ArtifactCacheLifetime	*number of seconds* Default: 30	The artifact cache lifetime in seconds.
ArtifactLifetime	*number of seconds* Default: 120	The number of seconds that artifacts are valid.
ArtifactResolutionServiceEndpoint	*URL*	URL of the artifact resolution service endpdoint.
ArtifactResolutionServiceList	*URL*	URL of the artifact resolution service endpdoint. **Note:** Only used for SAML 2.0
AssertionSigningKeyIdentifier	*keyname*	A special key used for signing a SAML assertion. Use this property only if you want to use different keys for the assertion and the SAML responses.
AssertionValidAfter	*number of seconds* Default: 60	The number of seconds that an assertion is considered valid after its issue date.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
AssertionValidBefore	*number of seconds* Default: 60	The number of seconds that an assertion is considered valid before its issue date.
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that format the rule, so that it can be placed in the XML response file. This mapping rule processes attribute query requests. Use this property if you do not want to specify a mapping rule in a file, or if you are modifying a federation. If you want to edit the XSLT rule as a regular file, use the **AttributeQueryMappingRuleFileName** property.
AttributeQueryMappingRuleFileName	*path and file name*	Specifies the path name to an XSLT file that is a mapping rule. This mapping rule processes attribute query requests. When defined, it takes precedence over the **AttributeQueryMappingRule** property.
AttributeAuthorityEnabled	`true` or `false` Default: `false`	Specifies whether the attribute query feature is activated in the federation. A value of `true` activates attribute query. A value of `false` disables attribute query.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name, such as `/saml20`, appended to it.
CommonDomainCookieLifetime	*number of seconds*	The number of seconds during which a cookie is active. -1 indicates that no expiration is defined. A positive value indicates the expiration time in seconds.
CommonDomainCookieReader	*URL*	The URL of the provider of the common domain service. It must be specified as a URL of the common domain service for the provider and must include the common domain value. The URL specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter or cdcreader appended to the end of the URL. This part of the URL is mandatory ifIBM Tivoli Federated Identity Manager is hosting the discovery service. If you are using a third party or custom discovery service, then that part of the URL is not required.For example, a system named `sp.example.com` with a common domain value of `somecommondomain.com` would have a URL with the following format: `https://sp.somecommondomain.com/FIM/sps/samlfed/saml20/cdcreader`

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
CommonDomainCookieWriter	*URL*	The URL of the provider of the common domain service. It must be specified as a URL of the common domain service for the provider and must include the common domain value. The URL specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter or cdcreader appended to the end of the URL. This part of the URL is mandatory if IBM Tivoli Federated Identity Manager is hosting the discovery service. If you are using a third party or custom discovery service, then that part of the URL is not required. For example, a system named `idp.example.com` with a common domain value of `somecommondomain.com` would have a URL with the following format: `https://idp.somecommondomain.com/FIM/sps/samlfed/saml20/cdcwriter`
CommonDomainName	*URL*	The name of the domain shared with other members of the federation. For example, `example.com`. There is no default value.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a Web site of the company that is associated with the federation.

ContactType	One of the supported contact types:
	`CONTACT_TYPE_ SUPPORT`
	`CONTACT_TYPE_ TECHNICAL`
	`CONTACT_TYPE_ ADMIN`
	`CONTACT_TYPE_ BILLING`
	`CONTACT_TYPE_ OTHER`
		The type of contact.

CreateMultipleAttributes	`true` or `false`	Setting that specifies whether multiple attribute statements are kept 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 `false`, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document. A value of `false` is appropriate for most configurations.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description

DefaultNameIDFormat	One of the supported values:
	`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
	or
	`urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
	or
	`urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
		Setting that specifies how a message with the unspecified name identifier is processed. **Note:** A partner-level setting takes precedence over any **DefaultNameIDFormat** set at the federation level. When no value is present at either the partner or federation level, the processing of unspecified name identifier is the same as a persistent name identifier.

EmailAddress	*e-mail address of the contact person*	The e-mail address of the contact person at the company that is associated with the federation.
EncryptionKeyIdentifier	*name of keystore and key*	The name of the encryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
EnhancedClientProxyEnabled	`true` or `false` Default: `false`	A setting that indicates whether a enhanced client proxy is enabled in the federation. A value of `true` enables the setting and a value of `false` disables the setting.
FedName	*name of federation*	The name of the federation you want to create or modify.
FirstName	*first name of a company contact person*	The first name of a contact person at the company that is associated with the federation.
ForceAuthn	`true` or `false`	Forces the identity provider to authenticate a user even when the user has previously authenticated. This option is specified only when configuring a service provider federation.
HttpHeaders	*header1,header2*	A comma separated list of HTTP headers that the identity provider can expect to see from the Enhanced Client Proxy clients containing identity information.
IncludeCertData	`yes` or `no`	Setting that specifies whether the certificate data should be provided with the signature.
IncludeIssuerDetails	`yes` or `no`	Setting that specifies whether the certificate issuer details should be provided with the signature.
IncludePublicKey	`yes` or `no`	Setting that indicates whether your public key should be included with the signature.
IncludeSubjectKeyId	`yes` or `no`	Setting that indicates whether the subject key identifier should be included with the signature.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
IncludeSubjectName	`true` or `false`	Setting that specifies whether the subject name should be provided with the signature.
IpDiscoveryEnabled	`true` or `false` Default: `false`	A setting that indicates whether identity provider discovery is enabled in the federation. A value of `true` enables the setting and a value of `false` disables the setting.
IsPassive	`true` or `false`	Setting that prevents the identity provider from interacting with principal (user).
LastName	*last name of the company contact person.*	The last name of the contact person at the company that is associated with the federation.
LogoutRequestLifetime	*number of seconds* Default: 120	The number of seconds that logout requests remain valid.
MapModuleInstanceId	`default-tdi` or *ID*	The identity mapping module instance, if using XSLT use default_map. To use Tivoli Directory Integrator, use `default-tdi`. **Note:** If you want to use a Tivoli Directory Integrator or custom mapping module, you must specify it when you create the response file; otherwise, the setup properties is not added to the response file. These properties are module dependent.
MappingRule	*content of the mapping rule file*	This property contains the actual mapping rule contents (XSL) that are needed for the rule to be properly formatted so that it can be contained in an XML file (the response file). Use this property if you do not want to point to a mapping rule in the file system or if you modify a federation. If you want to edit the XSLT rule as a regular file, you can do that and supply it to the response file using the **`MappingRuleFileName`** property.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system that is used as a mapping rule. It takes precedence over the **`MappingRule`** property if defined.
MsgLifetime	*number of seconds* Default: 300	The number of seconds that messages are valid.
NimIPArtifactEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier artifact binding. A value of `true` enables the binding and a value of `false` disables the binding.
NimIPPostEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier POST. A value of `true` enables the POST binding and a value of `false` disables POST binding.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
NimIPRedirectEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier redirect binding. A value of `true` enables the binding and a value of `false` disables the binding.
NimIPSOAPEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier SOAP. A value of `true` enables the SOAP binding and a value of `false` disables the SOAP binding.
NimReturnUrl	*URL*	Setting that indicates the endpoint where NameIdManagement Response messages are to be sent for the partner. A partner can specify this value for bindings that are asynchronous like HTTPRedirect and HTTPPost and HTTPArtifact.
NimSPArtifactEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier artifact binding. A value of `true` enables the binding and a value of `false` disables the binding.
NimSPPostEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier POST. A value of `true` enables the POST binding and a value of `false` disables POST binding.
NimSPRedirectEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier redirect binding. A value of `true` enables the binding and a value of `false` disables the binding.
NimSPSOAPEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier SOAP. A value of `true` enables the SOAP binding and a value of `false` disables the SOAP binding.
NimUrl	*URL*	The URL of the name identifier service. The value is the URL of your point of contact server with `/mnids` appended to it.
PartnerUsesBrowserPost	`true` or `false`	Setting that indicates whether your partner uses browser POST in the SAML 1.x federation.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Protocol	`SAML1_0`, `SAML1_1` , or `SAML2_0`	The protocol used by the federation.
ProtocolId	*URL*	Used in SAML 1.x federations; also referred to as a **`ProviderId`**. 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 saml_fed: `https://idp.example.com/sps/saml_fed/saml`

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
ProviderId	*URL of your point of contact server*	The URL of your point of contact server.
RequireConsentToFederate	`true` or `false` Default: `true`	A setting that indicates whether a user's consent is required to create the federation. A value of `true` enables the setting and a value of `false` disables the setting.
Role	`ip` or `sp`	The role of the federation.
SAML10AssertionIssuerName	*URL*	This property defines the endpoint for the assertion issuer. This endpoint is the same as the Protocol Id.
SAML10AssertionValidAfter	*number of seconds* Default: 60	The number of seconds that assertions are valid after its issue date.
SAML10AssertionValidBefore	*number of seconds* Default: 60	The number of seconds that assertions are valid before its issue date.
SAML10OneTimeAssertionEnforcement	`true` or `false`	Also known as one-time assertion use enforcement. Specifies that assertions can only be used one time.
SAML11AssertionIssuerName	*URL*	This property defines the endpoint for the assertion issuer. This endpoint is the same as the Protocol Id.
SAML11AssertionValidAfter	*number of seconds* Default: 60	The number of seconds that assertions are valid after its issue date.
SAML11AssertionValidBefore	*number of seconds* Default: 60	The number of seconds that assertions are valid before its issue date.
SAML11OneTimeAssertionEnforcement	`true` or `false`	Also known as one-time assertion use enforcement. Specifies that assertions can only be used one time.
SessionTimeout	*number of seconds* Default: 7200	The number of seconds that artifacts are valid.
SignArtifactRequest	`true` or `false`	Setting that indicates whether to sign incoming messages.
SignArtifactResponse	`true` or `false`	Setting that indicates whether to sign outgoing messages.
SignAttributeQueryRequest	*true or false*	Specifies whether the provider signs outgoing attribute query requests.
SignAttributeQueryResponse	*true or false*	Specifies whether the provider signs outgoing attribute query responses.
SigningKeyIdentifier or **SigningKeyId**	*name of keystore and key*	The name of the signing key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
SignLogoutRequest	`true` or `false`	Setting that indicates whether to sign logout requests.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
SignLogoutResponse	`true` or `false`	Setting that indicates whether to sign logout responses.
SignNameIdManagementRequest	`true` or `false`	Setting that indicates whether to sign the requests.
SignNameIdManagementResponse	`true` or `false`	Setting that indicates whether to sign the responses.
SignOnEndpoint	*URL*	Also known as the intersite transfer service URL. The URL to which the service provider sends authentication requests. A default value is provided. For example, `https://idp.example.com/sps/saml_fed/saml/login`.
SignSamlMsgs	`true` or `false` Default: `true`	Setting that indicates whether all the outgoing SAML messages and assertions must be signed. A value of `true` indicates that the response must be signed. A value of `false` indicates that the response does not have to be signed.
SignTypicalSamlMsgs	`true` or `false` Default: `true`	Setting that indicates whether the typical outgoing SAML message and assertion must be signed, except for ArtifactResponse and AuthnResponse. A value of `true` indicates that the response must be signed. A value of `false` indicates that the response does not have to be signed.
SloIPArtifactEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier artifact binding. A value of `true` enables the binding and a value of `false` disables the binding.
SloIPPostEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier POST. A value of `true` enables the POST binding and a value of `false` disables POST binding.
SloIPRedirectEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier redirect binding. A value of `true` enables the binding and a value of `false` disables the binding.
SloIPSOAPEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier SOAP. A value of `true` enables the SOAP binding and a value of `false` disables the SOAP binding. **Note:** This property is not supported in Tivoli Federated Identity Manager Business Gateway.
SloReturnUrl	*URL*	Setting that indicates the endpoint where LogoutResponse messages are to be sent for the partner. A partner can specify this value for bindings that are asynchronous such as HTTPRedirect and HTTPPost and HTTPArtifact.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
SloSPArtifactEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier artifact binding. A value of `true` enables the binding and a value of `false` disables the binding.
SloSPPostEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier POST. A value of `true` enables the POST binding and a value of `false` disables POST binding.
SloSPRedirectEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier redirect binding. A value of `true` enables the binding and a value of `false` disables the binding.
SloSPSOAPEnabled	`true` or `false` Default: `false`	Setting that enables or disables the name identifier SOAP. A value of `true` enables the SOAP binding and a value of `false` disables the SOAP binding. **Note:** This property is not supported in Tivoli Federated Identity Manager Business Gateway.
SloUrl	*URL*	The URL for single logout. The value is the URL of your point of contact server with `/slo` appended to it.
SoapEndpt	*URL of SOAP endpoint*	The URL of your SOAP endpoint. The value is the URL of your point of contact server with `/soap` appended to it. For example, `https://sp/FIM:9444/sps/responsefilesp/saml20/soap`.
SoapRequestClientBasicAuthPassword	*password*	The password that is used for client authentication.
SoapRequestClientBasicAuthUser	*username*	The username that is used for client authentication.
SoapRequestServerCertAuthKeyId	*name of keystore and key*	The name of the server validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
SsoArtifactEnabled	`true` or `false` Default: `false`	Setting that enables or disables the HTTP artifact binding for SAML 2.0 single sign-on. A value of `true` enables the binding and a value of `false` disables the binding.
SsoPostEnabled	`true` or `false` Default: `false`	Setting that enables or disables the HTTP POST binding for SAML 2.0 single sign-on. A value of `true` enables the binding and a value of `false` disables the binding.
SsoRedirectEnabled	`true` or `false` Default: `false`	Setting that enables or disables the HTTP redirect binding for SAML 2.0 single sign-on. A value of `true` enables the binding and a value of `false` disables the binding.
SsoUrl	*URL of the single sign-on service*	The URL of the single sign-on service. The value is the URL of your point of contact server with `/login` appended to it.

Table 20. Parameters in SAML federation response files (continued)

Parameter	Value	Description
UseClientBasicAuth	`true` or `false`	Setting that specifies whether basic client authentication is used.
UseSoapClientCertAuth	`true` or `false`	Setting that specifies whether client certificate authentication is used.
UseSoapServerCertAuth	`true` or `false`	Setting that specifies whether server certificate authentication is used.
ValidateTypicalSamlMsgs	`true` or `false`	Setting that indicates whether to validate signed messages.
WantAssertionSigned	`true` or `false` Default: `true`	Setting that indicates whether the incoming SAML message and assertion must be signed. A value of `true` indicates that the request must be signed. A value of `false` indicates that the request does not have to be signed.
WantAuthnRequestSigned	`true` or `false` Default: `true`	Setting that indicates whether the incoming SAML message and assertion must be signed. A value of `true` indicates that the request must be signed. A value of `false` indicates that the request does not have to be signed.
XsltMapping	`true` or `false`	A setting that indicates that XSLT is being used for mapping. A value of `true` indicates that XSLT is used and `false` indicates that a mapping module is used. If this parameter is set to `true`, specify values for either the **`MappingRule`** or the **`MappingRuleFileName`** properties. If it is set to `false`, values must be specified for the **`MapModuleInstanceId`**.

WS-Federation federation response file

Before you can create a federation using the **`manageItfimFederation`** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

You can create a response file for a new federation or one that is based on an existing federation.

New federation

Create a response file for creating a new federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -protocol type -role role -fileId output_file}
```

Existing federation

Create a response file for creating a federation that is based on an existing federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the protocol and federation role, either `identityProvider` or `serviceProvider`, that you specify in the command or that are specified by the properties of the existing federation input file.

The list that follows describes the parameters used in the response files.

Note: If you created a response file that is based on an existing federation, values are automatically specified for many of the parameters.

To edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

Examples of the response files are in the following directories:

AIX, Linux or Solaris

`/opt/IBM/FIM/examples/responsefiles`

Windows

`C:\Program Files\IBM\FIM\examples\responsefiles`

Parameters

Table 21. Parameters in WS-Federation federation response files

Parameter	Value	Description
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name, such as `/saml20`, appended to it.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a Web site of the company that is associated with the federation.
ContactType	`CONTACT_TYPE_TECHNICAL`	The only supported contact type is `CONTACT_TYPE_ TECHNICAL.`
EmailAddress	*e-mail address of the contact person*	The e-mail address of the contact person at the company that is associated with the federation.
Endpoint	*URL*	The URL of the endpoint for all requests for WS-Federation services.
FedName	*name of federation*	The name of the federation.

Table 21. Parameters in WS-Federation federation response files (continued)

Parameter	Value	Description
FirstName	*first name of a company contact person*	The first name of a contact person at the company that is associated with the federation.
LastName	*last name of the company contact person.*	The last name of the contact person at the company that is associated with the federation.
MapModuleInstanceId	`default-tdi` or *ID*	The identity mapping module instance, if using XSLT use default_map. To use Tivoli Directory Integrator, use `default-tdi`. **Note:** To use a Tivoli Directory Integrator or custom mapping module, specify it when you create the response file; otherwise, the setup properties will not be added to the response file. These properties are module dependent.
MappingRule	*content of the mapping rule file*	This property contains the actual mapping rule contents (XSL) that are needed for the rule to be properly formatted so that it can be contained in an XML file (the response file). Use this property if you do not want to point to a mapping rule in the file system or if you modify a federation. If you want to edit the XSLT rule as a regular file, you can do that and supply it to the response file using the **`MappingRuleFileName`** property.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system used as a mapping rule. It takes precedence over the **`MappingRule`** property if defined.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Protocol	Default: `WSFED`	The protocol used by the federation.
Realm	*URL*	The unique name of the WS-Federation Realm of your partner. The Realm name is included in assertions that are sent to federation providers. Providers rely on finding a known (defined) Realm name to accept the assertions.
Role	`ip` or `sp`	The role of the federation.
SAML11AssertionIssuerName	*URL*	The URL of the WS-FED issuer. This property has the same value as the Realm and BaseUrl properties.
SAML11AssertionValidAfter	*number of seconds* Default: 60	The number of seconds that assertions are valid after its issue date.

Table 21. Parameters in WS-Federation federation response files (continued)

Parameter	Value	Description
SAML11AssertionValidBefore	*number of seconds* Default: 60	The number of seconds that assertions are valid before its issue date.
SAML11OneTimeAssertionEnforcement	`true` or `false`	This is a property for the service provider. If set to `true`, assertions can only be used one time.

SecurityTokenId	**For SAML 1.1 module:**
	`default-saml1_1`
	For WSFed module:
	`default-wsfed`
		WS-Federation can use either a WSFed token module or SAML 1.1. Use this property to specify which module to use.

XsltMapping	`true` or `false` Default: `true`	A setting that indicates that XSLT is being used for mapping. A value of `true` indicates that XSLT is used and `false` indicates that a mapping module is used. If this parameter is set to `true`, specify values for either the **`MappingRule`** or the **`MappingRuleFileName`** properties. If it is set to `false`, values must be specified for the **`MapModuleInstanceId`**.

OpenID federation response file reference

Before you can create a federation using the **`manageItfimFederation`** command, you must create a response file. After creating the response file, edit the file so that it contains the appropriate values for your environment.

You can create a response file for a new federation or one that is based on an existing federation.

New federation

Create a response file for a new federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -protocol type -role role -fileId output_file}
```

Existing federation

Create a response file for a federation that is based on an existing federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the protocol and federation role.

The list that follows describes the parameters used in the response files.

Note: If you created a response file that is based on an existing federation, values are automatically specified for many of the parameters.

To edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

Parameters

Table 22. Parameters in OpenID federation response file for Service Providers

Parameter	Value	Description
AccessApprovalModuleID	*default value or custom value*	Setting that indicates if a custom dynamic endpoint access plug-in is used to check if specified endpoints can be trusted. If set to default access approval, only endpoints in the `allow` or `deny` lists are checked.
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
AllowedHosts	*hostname*	Specifies a list of regular expressions that identify host names to which the user agent can request access.
AllowedIps	*IP address*	Specifies a list of regular expressions that identify IP addresses or netmasks to which the user agent can request access.
AllowXriSupport	`true` or `false`	Specifies whether to resolve URL or XRI-based claimed identifiers. If you do not specify a value, the software uses only URL-based claimed identifiers.
AllowYadisDiscovery	`true` or `false`	Specifies whether to perform Yadis discovery.
AuthenticationMode	`checkid_immediate` or `checkid_setup`	Specifies the type of authentication mode to use.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name appended to.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a Web site of the company that is associated with the federation.
ContactType	One of the supported contact types: • `CONTACT_TYPE_ SUPPORT` • `CONTACT_TYPE_ TECHNICAL` • `CONTACT_TYPE_ ADMIN` • `CONTACT_TYPE_ BILLING` • `CONTACT_TYPE_ OTHER`	The type of contact.

Table 22. Parameters in OpenID federation response file for Service Providers (continued)

Parameter	Value	Description
DeniedHosts	*hostnames*	A list of regular expression strings denied for the hostname.
DeniedIps	*IP address*	Specifies list of denied IP addresses to which the user agent cannot request access.
DiscoveredInformationExpirationTime	*number of seconds*	Specifies the number of seconds the cache stores the discovered information.
EmailAddress	*e-mail address of the contact person*	The e-mail address of the contact person at the company that is associated with the federation.
FedName	*name of federation*	The name of the federation you want to create or modify.
FirstName	*first name of a company contact person*	The first name of a contact person at the company that is associated with the federation.
HttpAgentConnTimeout	*number of seconds*	Specifies the number of seconds that the HTTP session remains valid when there is no activity.
IvCredAttributeTypes	*types of attributes*	Specifies the types of attributes to include in the assertion.
LastName	*last name of the company contact person.*	The last name of the contact person at the company that is associated with the federation.
LoginEndpoint	*URL*	An endpoint for user login.
LoginReturnEndpoint	*URL*	Specifies the URL that the service provider requests the identity provider to send back the authentication response to.
MapModuleInstanceId	*identity mapping module instance*	The module that specifies mapping rules for identities.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system used as a mapping rule. It takes precedence over the **`MappingRule`** property if defined.
PapeAssuranceLevels	*namespace*	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.

Table 22. Parameters in OpenID federation response file for Service Providers (continued)

Parameter	Value	Description
PapeAuthenticationAge	*number of seconds*	Specifies how many seconds can lapse since the last authentication at the identity provider before prompting the user to authenticate again. If the user has not authenticated at the identity provider within the specified number of seconds, the identity provider must re-authenticate the user. If you enter a value equal to or less than zero, the identity provider is required to always re-authenticate the user.
PapeAuthPolicies	*URI*	Specifies a set of authentication policy URIs. The URIs represent authentication policies that the identity provider is required to satisfy when authenticating a user. If multiple policies are requested, the identity provider is required to satisfy as many of them as it can. The identity provider then indicates which authentication policies were satisfied in the response.
PapeEnabled	`true` or `false`	Specifies if the Provider Authentication Policy Extension attributes in the authentication request are sent to the identity provider.
PapeSkewTime	*number of seconds*	The number of seconds used to account for clock skew between: • the last authentication time returned by the identity provider mapping rule • the clock of the identity provider
PapeStrict	`true` or `false`	Specifies whether strict enforcement of PAPE is required or not.
PermittedServerProtocols	*HTTPS only, HTTP only, or both*	Specifies the protocols for the OpenID servers to which the user agent permits connection. The user agent can connect using: • HTTPS only • HTTP only • both HTTPS and HTTP
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.

Table 22. Parameters in OpenID federation response file for Service Providers (continued)

Parameter	Value	Description
PreferredVersion	*Can be left not set, but typically one of the following:* • `http://specs.openid.net/auth/2.0` (default) • `http://openid.net/signon/1.1`	If an identity provider supports both OpenID 1.1 and 2.0, this parameter determines whether or not the relying party attempts an OpenID 2.0 or 1.1 login. By default an OpenID 2.0 authentication is attempted and is kept unchanged for best practices.
Protocol	`OPENID`	The protocol used by the federation.
ProtocolId	*URL*	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. Also referred to as **`ProviderId`**.
ProviderId	*URL*	The URL of your point of contact server.
ResponseNonceSkewTime	*number of seconds*	Indicates how many seconds a relying party that is operating without an established association has, before they must perform the `check_authentication` request.
Role	`ip` or `sp`	The role of the federation.
TrustedKeyStore	*keystore* Default: `DefaultTrustedKeyStore`	Select the keystore containing the key or certificate to be used by the HTTPS client of the consumer. The keystore determines if the host that it is connected to can be trusted.
TrustedRoot	*URL*	Specifies the basis for trust shown to the user at the identity provider. The value is appended with a forward slash. For example, `https://webseald.example.com/`
UserAgentDenyConn	`true` or `false`	Specify to deny access to OpenID hosts by default.
XriProxies	*URL*	Specifies a list of URLs for resolving XRI identifiers. The URL must contain the `@XRI@` macro.

Table 22. Parameters in OpenID federation response file for Service Providers (continued)

Parameter	Value	Description
XsltMapping	`true` or `false`	A setting that indicates that XSLT is being used for mapping. A value of `true` indicates that XSLT is used and `false` indicates that a mapping module is used. If this parameter is set to `true`, specify values for either the **`MappingRule`** or the **`MappingRuleFileName`** properties. If it is set to `false`, values must be specified for the **`MapModuleInstanceId`**.

Table 23. Parameters in OpenID federation response file for Identity Providers

Parameter	Value	Description
AccessApprovalModuleID	*default value or custom value*	Setting that indicates if a custom dynamic endpoint access plug-in is used to check if specified endpoints can be trusted. If set to default access approval, only endpoints in the `allow` or `deny` lists are checked.
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
AllowedHosts	*hostname*	Specifies a list of regular expressions that identify host names to which the user agent can request access.
AllowesIps	*IP address*	Specifies a list of regular expressions that identify IP addresses or netmasks to which the user agent can request access.
AssociationExpiration	*number of seconds*	Specifies the lifetime of the association handle. The default value is 3600.
AssociationTypes	*Multi-valued. Can be on or more of the following:* • DH-SHA256 • DH-SHA1 • no encryption • plain text	Specifies a list of supported association session types for the OpenID identity provider. If a service provider attempts to establish an association that is not in this list, the association fails.
AuthenticationEndpoint	*URL*	An endpoint that forces users to authenticate if they have not already done so.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name.
CompanyName	*name of your company*	The name of the company that is associated with the federation.

Table 23. Parameters in OpenID federation response file for Identity Providers (continued)

Parameter	Value	Description
CompanyUrl	*URL of your company's Web site*	A URL for a Web site of the company that is associated with the federation.
ContactType	One of the supported contact types: • `CONTACT_TYPE_ SUPPORT` • `CONTACT_TYPE_ TECHNICAL` • `CONTACT_TYPE_ ADMIN` • `CONTACT_TYPE_ BILLING` • `CONTACT_TYPE_ OTHER`	The type of contact.
DeniedHosts	*hostnames*	Specifies a list of denied hostnames.
DeniedIps	*IP address*	Specifies list of denied IP addresses.
EmailAddress	*e-mail address of the contact person*	The e-mail address of the contact person at the company that is associated with the federation.
FedName	*name of federation*	The name of the federation you want to create or modify.
FirstName	*first name of a company contact person*	The first name of a contact person at the company that is associated with the federation.
HttpAgentConnTimeout	*number of seconds*	Specifies the number of seconds that the HTTP session remains valid when there is no activity.
IdentityMode	`true` or `false`	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`.
IdentityPattern	*URL*	Represents the regular expression on which identity URLs are matched for the federation. IBM Tivoli Federated Identity Manager replaces the `@ID@` part.
IdGeneratorModuleId	*id generator*	Specifies which ID generator creates a value that replaces the `@ID@` of an identity URL.
IPgeneratedClaimedPattern	*URL*	Represents the regular expression on which identity URLs are matched for the federation.IBM Tivoli Federated Identity Manager replaces the `@ID@` part.
LastName	*last name of the company contact person.*	The last name of the contact person at the company that is associated with the federation.
MapModuleInstanceId	*identity mapping module instance*	The module that specifies mapping rules for identities.

Table 23. Parameters in OpenID federation response file for Identity Providers (continued)

Parameter	Value	Description
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system used as a mapping rule. It takes precedence over the **MappingRule** property if defined.
MappingRuleType	*XSL or JavaScript* or *Tivoli Directory Integrator*, or *a custom mapping module instance*	Specify whether you want to achieve identity mapping through use of XSL or JavaScript transformations, Tivoli Directory Integrator, or through a custom mapping module solution.
PapeMaxAuthAgeOverride	`true` or `false`	Specifies if a user is always required to authenticate. If selected, the Maximum authentication age allowable clock skew is disabled. The default-selected value is disabled.
PapeSkewTime	*number of seconds*	The number of seconds used to account for clock skew between: • the last authentication time returned by the identity provider mapping rule • the clock of the identity provider
PerformRPDiscovery	`true` or `false`	Specifies whether to attempt relying party discovery.
PermittedServerProtocols	*HTTPS or HTTP*	Specifies the protocols for the OpenID servers to which the user agent permits connection. The user agent can connect using: • HTTPS only • HTTP only • both HTTPS and HTTP
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Protocol	`OPENID`	The protocol used by the federation.
ProtocolId	*URL*	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. Also referred to as **ProviderID**.
ProviderId	*URL*	The URL of your point of contact server.

Table 23. Parameters in OpenID federation response file for Identity Providers (continued)

Parameter	Value	Description
RequireRPDiscovery	`true` or `false`	Specifies if IBM 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.
ResponseNonceExpiration	*number of seconds* Default: 30	Indicates how many seconds a relying party that is operating without an established association has, before they must perform the `check_authentication` request.
Role	`ip` or `sp`	The role of the federation.
RPDiscoveryExpirationSeconds	*number of seconds*	Determines how many seconds to cache information discovered about relying parties. If you enter a value equal to or less than zero, information is never cached.
ServerEndpoint	*URL*	An endpoint for making an OpenID request.
SiteManagerURLEndpoint	*URL*	An endpoint for managing the set of trusted and untrusted consumer sites.
SupportOPIdentifier	`true` or `false`	Specifies if `identifier_select` is supported when a consumer initiates single sign-on. Use this option if an identity provider uses XRDS. If not selected, this option disables all other `identifier_select` options.
TrustedKeyStore	*keystore* Default: `DefaultTrustedKeyStore`	Select the keystore containing the key or certificate to be used by the HTTPS client of the consumer. The keystore determines if the host that it is connected to can be trusted.
TrustedSitesManagerModuleId	*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.
UserAgentDenyConn	`true` or `false`	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.

Table 23. Parameters in OpenID federation response file for Identity Providers (continued)

Parameter	Value	Description
UserSetupURL	*URL*	Specifies the URL 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.
XsltMapping	`true` or `false`	A setting that indicates that XSLT is being used for mapping. A value of `true` indicates that XSLT is used and `false` indicates that a mapping module is used. If this parameter is set to `true`, specify values for either the **`MappingRule`** or the **`MappingRuleFileName`** properties. If it is set to `false`, values must be specified for the **`MapModuleInstanceId`**.

OAuth 1.0 federation response file reference

Create a federation response file using the **`manageItfimFederation`** command and edit it with the appropriate values for your environment.

You can create a response file for a new federation or one that is based on an existing federation.

New federation

Create a response file for a new federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -protocol type -role role -fileId output_file}
```

Existing federation

Create a response file for a federation that is based on an existing federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the protocol.

The list that follows describes the parameters used in the response files.

Note: If you created a response file that is based on an existing federation, values are automatically specified for many of the parameters listed in this topic.

To edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.

3. Specify the type of federation that you want to create.
4. Save and close the file.

Parameters

Table 24. Parameters in OAuth 1.0 federation response file for Service Providers

Parameter	Value	Description
AccessEndpoint	*URL*	An endpoint used by the OAuth client to request a set of token credentials using the set of temporary credentials and a verification code.
AccessTokenMaxLifetime	*number of seconds* Default: 604800	The maximum length of time, in seconds, for an access token that is received from the OAuth server to be valid.
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
AuthorizeEndpoint	*URL*	An endpoint where the resource owner grants the client authorization to access the protected resource.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name appended to it.
ClientsManagerEndpoint	*URL*	An endpoint for managing the set of trusted clients.
ClientProviderModuleId	`OAuth10ClientProviderFedsImpl` or *custom client provider module instance ID*	The module ID that specifies the client provider. This property is dependent on the selected client provider.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a website of the company that is associated with the federation.
ContactType	One of the supported contact types: • `CONTACT_TYPE_ SUPPORT` • `CONTACT_TYPE_ TECHNICAL` • `CONTACT_TYPE_ ADMIN` • `CONTACT_TYPE_ BILLING` • `CONTACT_TYPE_ OTHER`	The type of contact.
EmailAddress	*e-mail address of the contact person*	The email address of the contact person at the company that is associated with the federation.
ExternalClientProviderConfig	*collection of key-value pairs*	This property specifies configuration data for your custom external client provider plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. `GUIXML` describes the configuration data for the external client provider extension module. This property uses a map of key-value pairs. You must add the `OAUTH.ExtClientProvider` prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example" on page 241.
FedName	*name of federation*	The name of the federation you want to create or modify.

Table 24. Parameters in OAuth 1.0 federation response file for Service Providers (continued)

Parameter	Value	Description
FirstName	*first name of a company contact person*	The given name of a contact person at the company that is associated with the federation.
LastName	*last name of the company contact person.*	The family name of the contact person at the company that is associated with the federation.
MapModuleInstanceId	*identity mapping module instance*	The module that specifies mapping rules for identities.
MappingRule	*content of the mapping rule file*	This property contains the actual mapping rule contents (XSL) that are needed for the rule to be properly formatted so that it can be contained in an XML file (the response file). Use this property if you do not want to point to a mapping rule in the file system or if you modify a federation. If you want to edit the XSLT rule as a regular file, you can do that and supply it to the response file using the **MappingRuleFileName** property.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system used as a mapping rule. It takes precedence over the **MappingRule** property if defined.
OAuthTokenCacheConfig	*collection of key-value pairs*	This property specifies configuration data for your custom token cache plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. `GUIXML` describes the configuration data for the token cache extension module. This property uses a map of key-value pairs. You must add the `OAUTH.TokenCache` prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example" on page 241.
OAuthTokenCacheModuleId	`OAuth10TokenCacheDMAPImpl` or *custom token cache module instance ID*	Selects the implementation class for an OAuth token cache.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Protocol	Default: `OAUTH1_0`	The protocol used by the federation.
ProtocolId	*URL*	A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the service provider URL. Optionally it can include a port number. It is also referred to as **ProviderId**.
ProviderId	*URL*	A URL uniquely identifying the provider to its partner provider. It is also referred to as **ProtocolId**.
RequestEndpoint	*URL*	An endpoint used by the client to obtain a set of temporary credentials.
Role	Default: `sp`	The role of the federation.

Table 24. Parameters in OAuth 1.0 federation response file for Service Providers (continued)

Parameter	Value	Description
SkewTime	*number of seconds* Default: 300	The number of seconds used to account for clock skew between the OAuth server and OAuth client.
TokenLifetime	*number of seconds* Default: 300	The maximum length of time, in seconds, for the set of temporary credentials and verification code to be valid.
TokenLength	*integer* Default: 20	The length of access tokens generated by the OAuth server.
TrustedClientsManagerConfig	*collection of key-value pairs*	This property specifies configuration data for your custom trusted clients manager plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. `GUIXML` describes the configuration data for the trusted clients manager extension module. This property uses a map of key-value pairs. You must add the `OAUTH.TrustedClientsManager` prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example."
TrustedClientsManagerModuleId	`TrustedClientsManagerCookieImpl`, `TrustedClientsManagerMemoryImpl`, `TrustedClientsManagerAutoApproveImpl`, or *custom trusted clients manager module instance ID*	Selects the implementation class for a trusted client manager. The implementation persists data concerning decisions made by a user during client authorization.
TwoLeggedEnabled	`true` or `false` Default: `false`	A setting that enables or disables the two-legged OAuth validation.
VerifierCodeLength	*integer* Default: 20	The length of the verification code from the OAuth server.
WAYFCookieLifetime	*number of seconds* Default: 86400	The maximum length of time, in seconds, for the *where are you from* cookie to be valid.
XsltMapping	`true` or `false` Default: `true`	A setting that indicates that XSLT is being used for mapping. A value of `true` indicates that XSLT is used and `false` indicates that a mapping module is used. If this parameter is set to `true`, specify values for either the **MappingRule** or the **MappingRuleFileName** properties. If it is set to `false`, values must be specified for the **MapModuleInstanceId**.

Example

In the following example, the **OAuthTokenCacheConfig** parameter is used to show the correct syntax of a key-value pair map.

```
<void method="put">
   <string>OAuthTokenCacheConfig</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <object class="java.util.HashMap">
      <void method="put">
```

```
      <string>OAUTH.TokenCache.JDBCProvider</string>
      <array class="java.lang.String" length="1">
       <void index="0">
        <string>jdbc/OAuthDB</string>
       </void>
      </array>
     </void>
     <void method="put">
      <string>OAUTH.TokenCache.CleanupInterval</string>
      <array class="java.lang.String" length="1">
       <void index="0">
        <string>300</string>
       </void>
     </array>
     </void>
    </object>
   </void>
  </object>
 </void>
```

OAuth 2.0 federation response file reference

Create a federation response file using the **manageItfimFederation** command and edit it with the appropriate values for your environment.

New federation

Create a response file for a new federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -protocol type -role role -fileId output_file}
```

Existing federation

Create a response file for a federation that is based on an existing federation by running the following command:

```
$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

After you have either of these commands, a response file is created. The content of the file depends on the protocol.

The list that follows describes the parameters used in the response files.

Note: If you created a response file that is based on an existing federation, values are automatically specified for many of the parameters.

To edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

Parameters

Table 25. Parameters in OAuth 2.0 federation response file for service providers

Parameter	Value	Description
AccessTokenLength	*length of the access token* Default: 20	The length of an access token that is issued from the authorization server.

Table 25. Parameters in OAuth 2.0 federation response file for service providers (continued)

Parameter	Value	Description
AccessTokenLifetime	*number of seconds* Default: 3600	The validity of an access token in seconds.
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
AllowPublicClientsTokenEndpoint	`true` or `false` Default: `false`	This property specifies whether OAuth clients without a secret can access the token endpoint.
AuthorizationCodeLength	*length of the authorization code* Default: 30	The length of an authorization code that is generated from the authorization server. This property is required if the **SupportedAuthorizationGrants** property includes the value of `AUTHORIZATION_CODE`.
AuthorizationCodeLifetime	*number of seconds* Default: 300	The validity of the authorization code in seconds. This property is required if the **SupportedAuthorizationGrants** property includes the value of `AUTHORIZATION_CODE`.
AuthorizationEndpoint	*URL*	An endpoint where the resource owner grants the OAuth client access to the protected resource. The authorization code and implicit grant are the only grant types that use this endpoint.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name appended to it.
ClientsManagerEndpoint	*URL*	An endpoint for managing the set of trusted clients.
ClientProviderModuleId	`OAuth20ClientProviderFedsImpl` or *client provider module instance ID*	The module ID that specifies the client provider. This property depends on the selected client provider.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a website of the company that is associated with the federation.
ContactType	One of the supported contact types: • `CONTACT_TYPE_ SUPPORT` • `CONTACT_TYPE_ TECHNICAL` • `CONTACT_TYPE_ ADMIN` • `CONTACT_TYPE_ BILLING` • `CONTACT_TYPE_ OTHER`	The type of contact.
EmailAddress	*e-mail address of the contact person*	The email address of the contact person in the company that is associated with the federation.
ExternalClientProviderConfig	*collection of key-value pairs*	This property specifies configuration data for your custom external client provider plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. `GUIXML` describes the configuration data for the external client provider extension module. This property uses a map of key-value pairs. You must add the `OAUTH20.ExtClientProvider` prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example" on page 246.
FedName	*name of federation*	The name of the federation you want to create or modify.

Table 25. Parameters in OAuth 2.0 federation response file for service providers (continued)

Parameter	Value	Description
FirstName	*first name of a company contact person*	The given name of a contact person in the company that is associated with the federation.
IssueRefreshToken	`true` or `false` Default: `false`	This property can be set either to issue a refresh token to the OAuth client or not. This property can be used only if the **SupportedAuthorizationGrants** property includes any of these values: `AUTHORIZATION_CODE` or `RESOURCE_OWNER_PASSWORD_CREDENTIALS`.
LastName	*last name of the company contact person*	The family name of the contact person in the company that is associated with the federation.
MapModuleInstanceId	*identity mapping module instance*	The module that specifies mapping rules for identities.
MappingRule	*content of the mapping rule file*	This property contains the actual mapping rule contents (XSL) that are needed for the rule to be properly formatted so that it can be contained in an XML file (the response file). Use this property if you do not want to point to a mapping rule in the file system or if you modify a federation. If you want to edit the XSLT rule as a regular file, you can do that and supply it to the response file using the **MappingRuleFileName** property.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system used as a mapping rule. It takes precedence over the **MappingRule** property if defined.
MaxAuthorizationGrantLifetime	*number of seconds* Default: 604800	The maximum duration of a grant where the resource owner authorized the OAuth client to access the protected resource. This property can be used with authorization code and resource owner password credentials grant types only. The value for this lifetime must be greater than the values specified for the **AuthorizationCodeLifetime** and **AccessTokenLifetime** properties. This property can be used only if the **SupportedAuthorizationGrants** property includes any of these values: `AUTHORIZATION_CODE` or `RESOURCE_OWNER_PASSWORD_CREDENTIALS`.
OAuthTokenCacheConfig	*collection of key-value pairs*	This property specifies configuration data for your custom token cache plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. `GUIXML` describes the configuration data for the token cache extension module. This property uses a map of key-value pairs. You must add the `OAUTH20.TokenCache` prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example" on page 246.
OAuthTokenCacheModuleId	`OAuth20TokenCacheDMAPImpl` or *token cache module instance ID*	The implementation class for an OAuth token cache.

Table 25. Parameters in OAuth 2.0 federation response file for service providers (continued)

Parameter	Value	Description
OAuthTokenTypeConfig	*collection of key-value pairs*	**Note:** Do not modify this parameter. This property specifies the configuration parameters for the supported token types. The default token type plug-in that IBM Tivoli Federated Identity Manager provides has no configuration requirement.
OAuthTokenTypeModuleId	`OAuth20TokenTypeHandlerBearerImpl`	**Note:** Do not modify this parameter. The type of access token an OAuth client uses to make protected resource requests.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Protocol	Default: `OAUTH2_0`	The protocol used by the federation.
ProtocolId	*URL*	A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the service provider URL. Optionally it can include a port number. It is also referred to as **ProviderId**.
ProviderId	*URL*	A URL uniquely identifying the provider to its partner provider. It is also referred to as **ProtocolId**.
RefreshTokenLength	*length of the refresh token* Default: 40	The length of a refresh token that is generated from the authorization server. This property can be used with authorization code and resource owner password credentials grant types only. This property is required if the **IssueRefreshToken** property is set to `true`.
Role	`sp`	The role of the federation.
SupportedAuthorizationGrants	`AUTHORIZATION_CODE`, `IMPLICIT_GRANT`, `CLIENT_CREDENTIALS`, or `RESOURCE_OWNER_PASSWORD_CREDENTIALS`	The authorization grant types your OAuth 2.0 federation supports. You can specify one or more values from the list of available grant types. If this property includes the value of `AUTHORIZATION_CODE`, the **AuthorizationCodeLength**, **AuthorizationCodeLifetime**, **IssueRefreshToken**, and **RefreshTokenLength** properties must be specified. If this property includes the value of `RESOURCE_OWNER_PASSWORD_CREDENTIALS`, the **IssueRefreshToken** and **RefreshTokenLength** properties must be specified.
TokenEndpoint	*URL*	An endpoint where the OAuth client exchanges an authorization grant for an access token and an optional refresh token. This endpoint is used in every authorization grant type except for the implicit grant.

Table 25. Parameters in OAuth 2.0 federation response file for service providers (continued)

Parameter	Value	Description
TrustedClientsManagerConfig	*collection of key-value pairs*	This property specifies configuration data for your custom trusted clients manager plug-in. **Note:** Values for this property are only required if your custom plug-in requires parameter configuration. GUIXML describes the configuration data for the trusted clients manager extension module. This property uses a map of key-value pairs. You must add the OAUTH20.TrustedClientsManager prefix to the key. The value can have multiple data. For an example of the key-value pairs map, see "Example."
TrustedClientsManagerModuleId	TrustedClientsManagerCookieImpl, TrustedClientsManagerMemoryImpl, TrustedClientsManagerAutoApproveImpl, or *custom trusted clients manager module instance ID*	Selects the implementation class for a trusted client manager. The implementation persists data concerning decisions made by a user during client authorization.
WAYFCookieLifetime	*number of seconds* Default: 86400	The maximum length of time, in seconds, for the *where are you from* cookie to be valid.
XsltMapping	true or false Default: true	A setting that indicates that XSLT is being used for mapping. A value of true indicates that XSLT is used and false indicates that a mapping module is used. If this parameter is set to true, specify values for either the **MappingRule** or the **MappingRuleFileName** properties. If it is set to false, values must be specified for the **MapModuleInstanceId**.

Example

In the following example, the **OAuthTokenCacheConfig** parameter is used to show the correct syntax of a key-value pair map.

```
<void method="put">
   <string>OAuthTokenCacheConfig</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <object class="java.util.HashMap">
      <void method="put">
       <string>OAUTH20.TokenCache.JDBCProvider</string>
       <array class="java.lang.String" length="1">
        <void index="0">
         <string>jdbc/OAuthDB</string>
        </void>
       </array>
      </void>
      <void method="put">
       <string>OAUTH20.TokenCache.CleanupInterval</string>
       <array class="java.lang.String" length="1">
        <void index="0">
         <string>300</string>
        </void>
      </array>
      </void>
     </object>
    </void>
   </object>
  </void>
```

manageItfimPartner

Use the **manageItfimPartner** command to manage a partner.

Purpose

The **manageItfimPartner** command can perform the following operations on a partner when used with the appropriate parameters:

- list
- create (using a response file)
- create the response file
- delete
- view
- enable
- disable
- modify

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimPartner {-operation operator -fimDomainName name [options]}
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-federationName name
-partnerName name
-partnerRole role
-mapModuleInstanceId ID
-fileId output_file | input_file
-signingKeystorePwd password
-encryptionKeystorePwd password
```

The use of these parameters depends on the operator you chose to use.

Parameters

The following parameters are available for use with the **manageItfimPartner** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform on the domain. Valid values are listed in the following table.

Table 26. Values for the manageItfimPartner -operation parameter

Value	Description and requirements
list	List all of the existing partners in all of the federations for a specific domain.

Table 26. Values for the manageItfimPartner -operation parameter (continued)

Value	Description and requirements

createResponseFile	Create a response file to create a partner in a federation. When you use this operator, you must also use the following parameters:
	
	federationName *name*
	This parameter specifies the name of the federation.
	
	fileId *output_name*
	This parameter specifies the name and path of the file that is created by the operator.
	
	partnerName *name*
	This parameter specifies the name of an existing partner on which you want to base the response file. Use this parameter only if you want to use details about an existing partner in the response file.
	
	If you want to create a response file for an attribute query request partner, you must add another parameter:
	
	partnerRole *role*
	For an attribute query request partner, the *role* value must be `qr`. You can optionally use this parameter for a service provider partner (`sp`) or identity provider partner (`ip`). When **partnerRole** is not specified, the command assigns the role based on the role of the federation.
	
	After you have created the response file, open it with a text editor. Review the attributes that are defined in the file and make the changes that are required by your environment.
	
	For information about the content of the response file, see:
	• "SAML partner response file reference" on page 253
	• "OAuth 1.0 partner response file reference" on page 266
	• "OAuth 2.0 partner response file reference" on page 268
	• "WS-Federation partner response file reference" on page 262
	
	Examples of the response files are in the following directories:
	
	AIX, Linux or Solaris
	`/opt/IBM/FIM/examples/responsefiles`
	
	Windows
	`C:\Program Files\IBM\FIM\examples\responsefiles`

Table 26. Values for the manageItfimPartner -operation parameter (continued)

Value	Description and requirements
create	Create a partner federation using a response file. When you use this operator, you must also use the following parameters: **federationName** *name* This parameter specifies the name of the federation. **fileId** *input_name* This parameter specifies the name and path of the response file that you are using as input. You can create the response file using the createResponseFile operator. **signingKeystorePwd** *password* The value used with this parameter is the password to the keystore where partner's signing key is stored. This is the key that you use to validate the partner's signature. **encryptionKeystorePwd** *password* The value used with this parameter is the password to the keystore where the encyrption key is stored. This is the key that you use to encrypt data to your partner. If you want to create an attribute query request partner, you must add another parameter: **partnerRole** *role* For an attribute query request partner, the *role* value must be `qr`. You can optionally use this parameter for a service provider partner (`sp`) or identity provider partner (`ip`). When **partnerRole** is not specified, the command assigns the role based on the role of the federation.
delete	Delete a partner. When you use this operator, you must also use the following parameters: **federationName** *name* This parameter specifies the name of the federation on which the operation occurs. **partnerName** *name* This parameter specifies the name of the partner on which the operation occurs.
view	View the details of a partner. When you use this operator, you must also use the following parameters: **federationName** *name* This parameter specifies the name of the federation on which the operation occurs. **partnerName** *name* This parameter specifies the name of the partner on which the operation occurs.

Table 26. Values for the manageItfimPartner -operation parameter (continued)

Value	Description and requirements
modify	Modify the properties of a partner. The use of this operation is a three-step process. You must first run the createResponseFile operator so that a file containing the properties in the federation are created. Next, you open the response file with a text editor to modify the properties you want to change. Save and close the file. Then, when you use the modify operator, you must specify the **fileId** ***name*** of the edited response file as part of the command. When you use this operator, you must use the following parameters: **federationName** *name* This parameter specifies the name of the federation on which the operation occurs. **partnerName** *name* This parameter specifies the name of the partner on which the operation occurs. **fileId** *name* This parameter specifies the name of the file that contains the properties for the partner.
enable	Enable a partner. When you use this operator, you must also use the following parameters: **federationName** *name* This parameter specifies the name of the federation on which the operation occursp. **partnerName** *name* This parameter specifies the name of the partner on which the operation occurs.
disable	Disable a partner. When you use this operator, you must also use the following parameters: **federationName** *name* This parameter specifies the name of the federation on which the operation occurs. **partnerName** *name* This parameter specifies the name of the partner on which the operation occurs.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name can be a string with characters of any type.

-federationName *name*
This parameter is required with all operators except **list**. The value used with this parameter is the name of the federation on which the operation is performed. The name can be a string with characters of any type.

-partnerName *name*
This parameter is required with all operators except **list**. The value used with this parameter is the name of the partner on which the operation is performed. The name can be a string with characters of any type.

-partnerRole *role*
This parameter is required if you want to create a response file for an attribute

query request partner or create an attribute query request partner. It is optional if you are managing service provider or identity provider partners. Supported values are:

ip Identity provider partner

sp Service provider partner

qr Attribute query request partner

If partnerRole is not specified, the command assigns the partner role to either service provider partner or identity provider partner, based on the federation provider type.

-mapModuleInstanceId *ID*
This parameter is required if you are creating a partner that uses a custom mapping module instead of an XSLT file for mapping. The ID that you specify is specific to the custom module you want to use.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file or creating a partner. The value used with this parameter is the file name and path of a response file that is read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

-signingKeystorePwd *password*
This parameter is required if you are creating a partner. The value used with this parameter is the password to the keystore where partner's signing key is stored. This is the key that you use to validate the partner's signature.

-encryptionKeystorePwd *password*
This parameter is required if you are creating a partner. The value used with this parameter is the password to the keystore where the encyrption key is stored. This is the key that you use to encrypt data to your partner.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the partners in a domain:

```
$AdminTask manageItfimPartner {-operation list -fimDomainName domain1}
```

Create a response file to create a partner:

```
$AdminTask manageItfimPartner {-operation createResponseFile
  -fimDomainName domain1 -federationName fed1
  -fileId c:\temp\saml2idp}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a partner. After you have created this file, open it with a text editor and define the attributes in it so that they are correct for your environment.

Create a response file based on an existing partner:

```
$AdminTask manageItfimPartner {-operation createResponseFile
  -fimDomainName domain1 -federationName fed1
  -partnerName idppartner -fileId c:\temp\saml2idp.rsp}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a partner or

modifying partner properties. After you have created this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a response file for an attribute query request partner

```
$AdminTask manageItfimPartner {-operation createResponseFile
  -fimDomainName domain1 -federationName fed1
  -partnerRole qr -fileId c:\temp\saml2idp.rsp}
```

Note: The file specified here is the name of the response file that you are creating with the command. You use this file as input for creating a partner or modifying partner properties. After you have created this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a partner:

```
$AdminTask manageItfimPartner {-operation create -fimDomainName domain1
  -federationName idpsaml2 -partnerName idppartner
  -fileId c:\temp\saml2idp -signingKeystorePwd testonly
  -encryptionKeystorePwd testonly}
```

Note: The file specified here is the response file and is used as input. Before running this command, you must have opened the response file with a text editor and ensured that the attributes that are defined in the file are correct for your environment.

Create an attribute query request partner

```
$AdminTask manageItfimPartner {-operation create -fimDomainName domain1
  -federationName idpsaml2 -partnerRole qr  -fileId c:\temp\saml2idp.rsp
  -signingKeystorePwd testonl  -encryptionKeystorePwd testonly}
```

Note: The file specified here is the response file. The command uses its contents as input. Before running this command, open the response file with a text editor and ensure that the attributes that are defined in the file are correct for your environment.

Delete a partner:

```
$AdminTask manageItfimPartner {-operation delete -fimDomainName domain1
  -federationName fed1 -partnerName idppartner}
```

View partner details:

```
$AdminTask manageItfimPartner {-operation view -fimDomainName domain1
  -federationName fed1 -partnerName idppartner}
```

Modify partner properties:

First run createResponseFile, as described in "Create a response file based on an existing partner," to create a file that contains all of the partner properties. Edit it using a text editor. Save it Then, reload it into your environment using the modify command.

```
$AdminTask manageItfimPartner {-operation modify -fimDomainName domain1
  -partnerName idppartner -fileId c:\temp\saml2idp.xml }
```

Enable a partner:

```
$AdminTask manageItfimPartner {-operation enable -fimDomainName domain1
  -federationName fed1 -partnerName idppartner}
```

Disable a partner:

```
$AdminTask manageItfimPartner {-operation disable -fimDomainName domain1
  -federationName fed1 -partnerName idppartner}
```

SAML partner response file reference

Before you can create a partner using the **manageItfimPartner** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

You can create a response file for creating a new partner by running the following command:

New partner

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

Existing partner

Create a response file for creating a partner that is based on an existing partner by running the following command:

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -partnerName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the federation role, either `identityProvider` or `serviceProvider`, specified in the command or by the properties of the existing partner.

Note: If you created a response file that is based on an existing partner, values are automatically specified for many of the parameters shown here.

Open the file with a text editor, review the attributes that are defined in the file, make changes to specify the appropriate values, and then save and close the file.

Examples of the response files are in the following directories:

AIX, Linux or Solaris

```
/opt/IBM/FIM/examples/responsefiles
```

Windows

```
C:\Program Files\IBM\FIM\examples\responsefiles
```

Parameters

Table 27. Parameters in SAML partner response files

Parameter	Value	Description
AllowIBMProtocolExtension	`true` or `false`	Setting that specifies whether the use of the IBM Protocol Extension is allowed or not. The extension allows a query-string parameter that specifies whether browser artifact or browser POST is used in a SAML 1.x federation.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
AnonymousUserUserName	*username*	This is a one-time 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 recognizes 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 is useful in scenarios where the service provider does not have 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. **Note:** The user identity must exist as a valid user in the user registry.
ArtifactCacheLifetime	*number of seconds*	The artifact cache lifetime in seconds. The default value is 30.
AssertionAttributeTypes	* or *specific_attribute_names*	Setting that indicates the types of attributes to include in the assertion. * is the default and indicates that all available attributes must be included.

AssertionDigestAlgorithm	**For SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#sha1`
	
	For SHA256
	
	`http://www.w3.org/2001/04/xmlenc#sha256`
	
	For SHA512
	
	`http://www.w3.org/2001/04/xmlenc#sha512`
		The digest algorithm used for signing the SAML 2 assertion. Use this property only if you want to use a different digest algorithm for the assertion from that of the SAML 2 message.

AssertionSigningKeyIdentifier	*ID*	A special key used for signing a SAML assertion. Use this property only if you want to use different keys for the assertion and the SAML responses.

AssertionSignatureAlgorithm	**For DSA-SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	For RSA-SHA1
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	For RSA-SHA256
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		The signature algorithm used for signing the SAML 2 assertion. Use this property only if you want to use a different signature algorithm for the assertion from that of the SAML message.
		Note: The signing key type must match the signature algorithm to successfully sign the SAML 2 assertion.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
AssertionValidateKeyIdentifier	*ID*	Setting that specifies the key to use to validate signatures. Supply the value in the format `keystoreName_KeyAlias`.
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that format the rule, so that it can be placed in the XML response file. The provider uses this mapping rule to process attribute query requests. Use this property if you do not want to specify a mapping rule in a file or if you are modifying a federation. If you want to edit the XSLT rule as a regular file, use the response file property **AttributeQueryMappingRuleFileName**.
AttributeQueryMappingRuleFileName	*path and file name*	Specifies path name to an XSLT file that is used as a mapping rule. The provider uses this mapping rule to process attribute query requests. When defined, it takes precedence over the **AttributeQueryMappingRule** property.

BlockEncAlgorithm	**For AES-128**
	
	`http://www.w3.org/2001/04/xmlenc#aes128-cbc`
	
	For AES-256
	
	`http://www.w3.org/2001/04/xmlenc#aes256-cbc`
	
	For AES-192
	
	`http://www.w3.org/2001/04/xmlenc#aes192-cbc`
	
	For Triple DES
	
	`http://www.w3.org/2001/04/xmlenc#tripledes-cbc`
		Setting that specifies the encryption algorithm that is used to encrypt data for a service provider partner.

ClientBasicAuth	`true` or `false`	Setting that indicates whether the partner requires authentication with a user name and password. `true` indicates a user name and password are required. When set to `true`, values must be specified for **ClientBasicAuthPwd** and **ClientBasicAuthUser**. `false` indicates a user name and password are not required. `false` is the default.
ClientBasicAuthPwd	*password*	The password that you must use to authenticate to the partner site.
ClientBasicAuthUser	*username*	The user name you must use to authenticate to the partner site.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
ClientCertAuth	`true` or `false`	Setting that indicates whether the partner requires authentication with a certificate. `true` indicates a certificate is required and is the default value. `false` indicates a user name and password are not required. Supply the value in the format `keystoreName_KeyAlias`.
ClientCertKeyId	*key_alias*	The name of the client certificate key.
CommonDomainCookieReader	*URL*	The URL of the provider of the common domain service. It must be specified as a URL of the common domain service for the provider and must include the common domain value. The URL specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter or cdcreader appended to the end of the URL. This part of the URL is mandatory if IBM Tivoli Federated Identity Manager is hosting the discovery service. If you are using a third party or custom discovery service, then that part of the URL is not required. For example, a system named `sp.example.com` with a common domain value of `somecommondomain.com` can have a URL with the following format: `https://sp.somecommondomain.com/FIM/sps/samlfed/saml20/cdcreader`
CommonDomainCookieWriter	*URL*	The URL of the provider of the common domain service. It must be specified as a URL of the common domain service for the provider and must include the common domain value. The URL specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter or cdcreader appended to the end of the URL. This part of the URL is mandatory if IBM Tivoli Federated Identity Manager is hosting the discovery service. If you are using a third party or custom discovery service, then that part of the URL is not required. For example, a system named `idp.example.com` with a common domain value of `somecommondomain.com` can have a URL with the following format: `https://idp.somecommondomain.com/FIM/sps/samlfed/saml20/cdcwriter`

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
CreateMulitipleAttributes	true or false	Setting that specifies whether multiple attribute statements are kept 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 false, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document. False is appropriate for most configurations.
DefaultNameIDFormat	One of the supported values: urn:oasis:names:tc:SAML:2.0:nameid-format:persistent or urn:oasis:names:tc:SAML:2.0:nameid-format:transient or urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress	Setting that specifies how a message with the unspecified name identifier is processed. **Note:** A partner-level setting takes precedence over any DefaultNameIDFormat set at the federation level. When no value is present at either the partner or federation level, the processing of unspecified name identifier is the same as a persistent name identifier.
DefaultPostAuthTargetURL	*URL*	The default URL to present to the user after authentication.
DigestAlgorithm	**For SHA1** http://www.w3.org/2000/09/xmldsig#sha1 **For SHA256** http://www.w3.org/2001/04/xmlenc#sha256 **For SHA512** http://www.w3.org/2001/04/xmlenc#sha512	Setting that specifies the digest algorithm that is used to sign the SAML 2 message for the partner. The same digest algorithm is also used to sign the SAML2 assertion when the **AssertionDigestAlgorithm** property is missing.
EncryptAssertion	true or false	Setting that specifies whether the name identifiers are encrypted. true means they are encrypted and false means they are not.
EncryptAssertionAttrs	true or false	Setting that specifies whether all the attributes are encrypted. true means they are encrypted and false means they are not.
encryptionKeyAlias	*key_alias_name*	The name of the key that is used to encrypt data that you send to your partner.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
EncryptionKeyIdentifier	*name of keystore and key*	The name of the encryption key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
encryptionKeystore	*keystore_name*	The name where the encryption key supplied in the partner metadata is stored. This property is not required if you are not using a partner metadata but is using a response file with all the properties defined. However, you must set the **EncryptionKeyIdentifier** to a proper value. The encryption or decryption key must also be in the keystore.
EncryptNameId	`true` or `false`	A setting that specifies whether the name identifiers is encrypted. `true` means they are encrypted and `false` means they are not.
IncludeCertData	`true` or `false`	A setting that indicates whether the certificate data is included with the signature.
IncludeInclusiveNamespaces	`true` or `false`	A setting that indicates whether the namespaces is included with the signature.
IncludeFedIdInAliasServiceLookup	`true` or `false`	A setting that indicates if the federation ID is combined with the partner Provider ID as a key into the alias service when performing alias service operations. Always set this value to `true`, unless you have aliases that were created for a partner before IBM Tivoli Federated Identity Manager 6.2.2 and have not migrated your alias service to the new format.
IncludeIssuerDetails	`true` or `false`	A setting that indicates whether the issuer details is included with the signature.
IncludePublicKey	`true` or `false`	A setting that indicates whether the public key is included with the signature.
IncludeSubjectKeyId	`true` or `false`	A setting that indicates whether the subject key identifier is included with the signature.
IncludeSubjectName	`true` or `false`	A setting that indicates whether the subject name is included with the signature.
IncludeX509CertData or **IncludeX509CertificateData**	`true` or `false`	A setting that indicates whether the certificate data is included with the signature.
IncludeX509IssuerDetails	`true` or `false`	A setting that indicates whether the issuer details is included with the signature.
IncludeX509SubjectKeyIdentifier	`true` or `false`	A setting that indicates whether the subject key identifier is included with the signature.
IncludeX509SubjectName	`true` or `false`	A setting that indicates whether the subject name is included with the signature.
LogoutRequestLifetime	*number of seconds*	The number of seconds that logout requests remain valid. The default value is 120.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
MapUnknownAlias	`true` or `false`	A setting that specifies that the service provider can map an unknown persistent name identifier alias to the anonymous user account. If set to `false` (the default value), unknown aliases are rejected.
metadataFileName	*file name of partner metadata file*	The file name of the partner's metadata file.
PartnerUsesBrowserPost	`true` or `false`	Setting that indicates whether your partner use browser POST in the SAML 1.x federation.
ProtocolId	*URL*	(Used in SAML 1.x federations; also referred to as a ProviderId). 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 saml_fed: `https://idp.example.com/sps/saml_fed/saml`
Role	`ip` or `sp`	The role of the federation.

SAML10AssertionSignatureAlgorithm	**For DSA-SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	For RSA-SHA1
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	For RSA-SHA256
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		The signature algorithm used for signing the SAML 1.0 assertion. Use this property only if you want to use a different signature algorithm for the assertion from that of the SAML message. **Note:** The signing key type must match the signature algorithm to successfully sign the SAML 1.0 assertion.

SAML11AssertionSignatureAlgorithm	**For DSA-SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	For RSA-SHA1
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	For RSA-SHA256
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		The signature algorithm used for signing the SAML 1.1 assertion. Use this property only if you want to use a different signature algorithm for the assertion from that of the SAML message. **Note:** The signing key type must match the signature algorithm to successfully sign the SAML 1.1 assertion.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
SamlExtendedAttributeTypes	* or *attribute_type*	Specifies 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 is included in the assertion. You can specify one or more attribute types individually. For example, if you want to include only attributes of type `urn:oasis:names:tc:SAML:2.0:assertion`, specify that value instead of *.
ServerCertKeyId	*key_alias*	The name of the server certificate key.
SessionTimeout	*number of seconds*	The number of seconds that artifacts are valid. The default value is 7200.
signArtifactRequest	`true` or `false`	A setting that indicates whether the requests are signed.
signArtifactResponse	`true` or `false`	A setting that indicates whether the responses are signed.
SignAttributeQueryResponse	*true or false*	Specifies whether the partner signs outgoing responses.
SignAttributeQueryRequest	*true or false*	Specifies whether the partner signs outgoing requests.

SignatureAlgorithm	**For DSA-SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	For RSA-SHA1
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	For RSA-SHA256
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		Setting that specifies the signature algorithm that is used to sign the SAML messages for the partner.
		
		The behavior differs depending on the SAML version that you have:
		
		For **SAML 1.0**, the same signature algorithm is also used to sign the SAML 1.0 assertion when the **SAML10AssertionSignatureAlgorithm** property is missing.
		
		For **SAML 1.1**, the same signature algorithm is also used to sign the SAML 1.1 assertion when the **SAML11AssertionSignatureAlgorithm** property is missing.
		
		For **SAML 2.0**, the same signature algorithm is also used to sign the SAML 2 assertion when the **AssertionSignatureAlgorithm** property is missing.
		Note: The signing key type must match the signature algorithm to successfully sign the SAML assertion.

signatureKeyAlias	*key_alias_name*	The name of the key that is used to validate message signatures.
signatureKeystoreName	*keystore_name*	The name of the keystore where the validation key is stored.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
SigningKeyIdentifier or **SigningKeyId**	*name of keystore and key*	The name of the signing key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
SignLogoutRequest	`true` or `false`	Setting that indicates if logout requests are signed.
signNameIdManagementResponse	`true` or `false`	A setting that indicates whether the responses are signed.
SignOnEndpoint	*URL*	Also known as the intersite transfer service URL. The URL to which the service provider sends authentication requests. A default value is provided. For example: `https://idp.example.com/sps/saml_fed/saml/login`
SignTypicalSamlMsgs	`true` or `false`	Setting that indicates whether the typical outgoing SAML message and assertion must be signed. `true` indicates that the response must be signed and is the default. `false` indicates that the response does not have to be signed.
SignLogoutResponse	`true` or `false`	Setting that indicates if logout responses are signed.
SoapRequestClientBasicAuthPassword	*password*	The password that is used for client authentication.
SoapRequestClientBasicAuthUser	*username*	The username that is used for client authentication.
SoapRequestClientCertAuthKeyId	*name of keystore and key*	The name of the client authentication key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
SoapRequestServerCertAuthKeyId	*name of keystore and key*	The name of the server validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
UseClientBasicAuth	`true` or `false`	Setting that specifies whether basic client authentication is used.
UseSoapClientCertAuth	`true` or `false`	Setting that specifies whether client certificate authentication is used.
UseSoapServerCertAuth	`true` or `false`	Setting that specifies whether server certificate authentication is used.
ValidateArtifactRequest	`true` or `false`	Setting that indicates whether the signatures in requests is validated. `true` indicates that the signatures is validated. `false` indicates that signatures is not validated and is the default value.
ValidateArtifactResponse	`true` or `false`	Setting that indicates whether the signatures in responses is validated. `true` indicates that the signatures is validated. `false` indicates that signatures is not validated and is the default value.

Table 27. Parameters in SAML partner response files (continued)

Parameter	Value	Description
ValidateAttributeQueryResponse	`true` or `false`	Specifies whether the provider validates the partner signatures on attribute query responses that it receives.
ValidateAttributeQueryRequest	`true` or `false`	Specifies whether the provider validates the partner signatures on attribute query requests that it receives.
ValidateKeyIdentifier	`true` or `false`	A setting that specifies the key to use to validate the signatures.
ValidateLogoutResponse	`true` or `false`	A setting that indicates that a logout response received from the partner signature is validated. An error occurs if the message is not signed.
ValidateLogoutRequest	`true` or `false`	A setting that indicates that a logout request received from the partner signature is validated. An error occurs if the message is not signed.
ValidateNameIdManagementResponse	`true` or `false`	A setting that indicates that a manage name id response received from the partner signature is validated. An error occurs if the message is not signed.
ValidateNameIdManagementRequest	`true` or `false`	A setting that indicates that a manage name id request received from the partner signature is validated. An error occurs if the message is not signed.
ValidateSamlMsgs	`true` or `false`	A setting that indicates whether the signatures in messages is validated. `true` indicates that the signatures in messages is validated and is the default value. `false` indicates that messages signatures is not validated.
ValidateTypicalSamlMsgs	`true` or `false`	A setting that indicates whether you validate signed messages.
ValidationKey	*name of keystore and key*	The name and keystore of the signature validation key that is used to validate messages from the partner.

WS-Federation partner response file reference

Before you can create a partner using the **manageItfimPartner** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

You can create a response file for creating a new partner by running the following command:

New partner

```
$AdminTask manageItfimPartner {-operation createResponseFile --fimDomainName name
  -federationName name -fileId output_file}
```

Existing partner

Create a response file for creating a partner that is based on an existing partner by running the following command:

```
$AdminTask manageItfimPartner {-operation createResponseFile --fimDomainName name
  -federationName name -partnerName name -fileId output_file}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the federation role, either `identityProvider` or `serviceProvider`, specified in the command or by the properties of the existing partner.

Note: If you created a response file that is based on an existing partner, values are automatically specified for many of the parameters shown here.

Open the file with a text editor, review the attributes that are defined in the file, make changes to specify the appropriate values, and then save and close the file.

Examples of the response files are in the following directories:

AIX, Linux or Solaris

`/opt/IBM/FIM/examples/responsefiles`

Windows

`C:\Program Files\IBM\FIM\examples\responsefiles`

Parameters

Table 28. Parameters in WS-Federation partner response files

Parameter	Value	Description
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
BaseUrl	*URL*	The URL of the point of contact server with the federation name and the protocol name, such as /saml20, appended to it.
CompanyName	*name of your company*	The name of the company that is associated with the federation.
CompanyUrl	*URL of your company's Web site*	A URL for a Web site of the company that is associated with the federation.
ContactType	`CONTACT_TYPE_TECHNICAL`	The only supported contact type is `CONTACT_TYPE_ TECHNICAL.`
EmailAddress	*e-mail address of the contact person*	The e-mail address of the contact person at the company that is associated with the federation.
Endpoint	*URL*	The URL of the endpdoint for all requests for WS-Federation services.
FirstName	*first name of a company contact person*	The first name of a contact person at the company that is associated with the federation.
IncludePublicKey	`true` or `false`	A setting that indicates whether the public key should be included with the signature.
IncludeX509CertificateData	`true` or `false`	A setting that indicates whether the certificate data should be included with the signature.

Table 28. Parameters in WS-Federation partner response files (continued)

Parameter	Value	Description
IncludeX509IssuerDetails	`true` or `false`	A setting that indicates whether the issuer details should be included with the signature.
IncludeX509SubjectKeyIdentifier	`true` or `false`	A setting that indicates whether the subject key identifier should be included with the signature.
IncludeX509SubjectName	`true` or `false`	A setting that indicates whether the subject name should be included with the signature.
LastName	*last name of the company contact person.*	The last name of the contact person at the company that is associated with the federation.
MapModuleInstanceId	`default-tdi` or *ID*	The identity mapping module instance, if using XSLT use default_map. To use Tivoli Directory Integrator, use `default-tdi`. **Note:** If you want to use a Tivoli Directory Integrator or custom mapping module, you need to specify it when you create the response file; otherwise, the setup properties are not added to the response file. These properties are module dependent.
MappingRule	*content of the mapping rule file*	This property contains the actual mapping rule contents (XSL) that are needed for the rule to be properly formatted so that it can be contained in an XML file (the response file). Use this property if you do not want to point to a mapping rule in the file system or if you are modifying a federation. If you want to edit the XSLT rule as a regular file, you can do that and supply it to the response file using the MappingRuleFileName property.
MappingRuleFileName	*path and file name*	This property points to an XSLT file in the file system that are used as a mapping rule. It takes precedence over the MappingRule property if defined.
MaxRequestLifetime	*number in seconds*	The maximum length of time, in seconds, that a request or message that is received from a WS-Federation partner is valid. When not specified, a value of -1 is used. A value of -1 means that the requests are not checked for validity.

Table 28. Parameters in WS-Federation partner response files (continued)

Parameter	Value	Description
PartnerConfig	`true` or `false`	The value is automatically specified when the response file is created. `true` specifies that the file is a partner response file. `false` specifies that the file is a federation response file.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the federation.
Realm	*URL*	The unique name of the partner's WS-Federation Realm. The Realm name is included in assertions that are sent to federation providers. Providers rely on finding a known (defined) Realm name to accept the assertions.
Role	`ip` or `sp`	The role of the federation.
SAML11CreateMulitipleUniversal UserAttributes	`true` or `false`	Setting that specifies whether multiple attribute statements are kept 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 `false`, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document. `False` is appropriate for most configurations.
SAML11ExtendedAttributeTypes	`*` or *specific_attribute_names*	Setting that indicates the types of attributes to include in the assertion. `*` is the default and indicates that all available attributes should be included.
SAML11IncludeInclusiveNamespaces	`true` or `false`	A setting that indicates whether the namespaces should be included with the signature.

SAML11AssertionSignatureAlgorithm	**For DSA-SHA1**
	
	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	For RSA-SHA1
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	For RSA-SHA256
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		The signature algorithm used for signing the SAML assertion. Use this property only if you want to use a different signature algorithm for the assertion from that of the SAML message.
		Note: The signing key type must match the signature algorithm to successfully sign the SAML assertion.

Table 28. Parameters in WS-Federation partner response files (continued)

Parameter	Value	Description
SAML11SigningKeyIdentifier	*name of keystore and key*	The name of the signing key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`, for example, `DefaultKeyStore_testkey`.
SAML11ValidationKey	*name of keystore and key*	The name of the validation key and the keystore in which it is stored. The names must be provided in the following format: `keystorename_keyname`.
XsltMapping	`true` or `false`	Setting that indicates that XSLT is being used for mapping. `true` indicates that XSLT is used and `false` indicates that a mapping module is used instead. If this is set to `true`, values must be specified for either the the MappingRule or the MappingRuleFileName properties. If it is set to `false`, values must be specified for the MapModuleInstanceId.

OAuth 1.0 partner response file reference

Create a partner response file using the **manageItfimPartner** command and edit it with the appropriate values for your environment.

Note: These partner response file parameters are only relevant for federations which are using IBM Tivoli Federated Identity Manager as the client configuration provider.

You can create a response file for creating a new partner by running the following command:

New partner

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

Existing partner

Create a response file for creating a partner that is based on an existing partner by running the following command:

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -partnerName name -fileId output_file}
```

After you have run either of these commands, a response file is created.

Note: If you created a response file that is based on an existing partner, values are automatically specified for many of the parameters.

To edit a response file:

1. Open the response file with a text editor.

2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

Parameters

Table 29. Parameters in OAuth 1.0 partner response files

Parameter	Value	Description
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
ClientCallbackURI	*URI*	A URI where the resource owner is redirected when authorization is completed. Set this value to `oob` if the OAuth client does not register a callback URI.
ClientIdentifier	*client key*	A unique identifier supplied to the OAuth client to identify itself to the OAuth server.
ClientSecret	*client secret*	A shared secret between the OAuth client and OAuth server used for signing requests.
CompanyName	*name of your company*	The name of the company that is associated with the partner.
CompanyUrl	*URL of your company's Web site*	A URL for a website of the company that is associated with the partner.
ContactType	One of the supported contact types: • `CONTACT_TYPE_ SUPPORT` • `CONTACT_TYPE_ TECHNICAL` • `CONTACT_TYPE_ ADMIN` • `CONTACT_TYPE_ BILLING` • `CONTACT_TYPE_ OTHER`	The type of contact.
EmailAddress	*e-mail address of the contact person*	The email address of the contact person at the company that is associated with the partner.
FirstName	*first name of a company contact person*	The given name of a contact person at the company that is associated with the partner.
LastName	*last name of the company contact person.*	The family name of the contact person at the company that is associated with the partner.

Table 29. Parameters in OAuth 1.0 partner response files (continued)

Parameter	Value	Description
OverrideCallbackURI	`true` or `false`	A setting that specifies whether the callback URI parameter in the request for a temporary credential overrides the value of the **ClientCallbackURI** property. true indicates that the OAuth client can override the configured client callback URI with the callback URI in the request for a temporary credential. false indicates that both the temporary credential callback URI and configured client callback URI match. The default value is set to false.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the partner.
Role	Default: `oc`	The role of the partner. `oc` stands for OAuth client.

OAuth 2.0 partner response file reference

Create a partner response file using the **manageItfimPartner** command and edit it with the appropriate values for your environment.

Note: These partner response file parameters are only relevant for federations which are using IBM Tivoli Federated Identity Manager as the client configuration provider.

You can create a response file for creating a new partner by running the following command:

New partner

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -fileId output_file}
```

Existing partner

Create a response file for creating a partner that is based on an existing partner by running the following command:

```
$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name
  -federationName name -partnerName name -fileId output_file}
```

After you have run either of these commands, a response file is created.

Note: If you created a response file that is based on an existing partner, values are automatically specified for many of the parameters.

To edit a response file:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.

4. Save and close the file.

Parameters

Table 30. Parameters in OAuth 2.0 partner response files

Parameter	Value	Description
AdditionalInfo	*additional information about the contact or company*	Any additional information about the contact or the company that you want to record.
ClientIdentifier	*client key*	A unique identifier supplied to the OAuth client to identify itself to the authorization server.
ClientSecret	*client secret*	A shared secret between the OAuth client and authorization server used for client authentication. If you do not register a secret for the OAuth client , it becomes a public client. If you do register a secret, the OAuth client becomes a confidential client.
CompanyName	*name of your company*	The name of the company that is associated with the partner.
CompanyUrl	*URL of your company's Web site*	A URL for a website of the company that is associated with the partner.
ContactType	One of the supported contact types: • CONTACT_TYPE_ SUPPORT • CONTACT_TYPE_ TECHNICAL • CONTACT_TYPE_ ADMIN • CONTACT_TYPE_ BILLING • CONTACT_TYPE_ OTHER	The type of contact.
EmailAddress	*e-mail address of the contact person*	The email address of the contact person at the company that is associated with the partner.
FirstName	*first name of a company contact person*	The given name of a contact person at the company that is associated with the partner.
LastName	*last name of the company contact person.*	The family name of the contact person at the company that is associated with the partner.
PhoneNumber	*phone number of the contact person*	The phone number of the contact person at the company that is associated with the partner.
RedirectionURI	*URI*	A URI where the resource owner is redirected when authorization is completed. If you do not want to register a redirection URI, leave this property empty.
Role	Default: oc	The role of the partner. oc stands for OAuth client.

manageItfimPointOfContact

Use the **manageItfimPointOfContact** command to manage a custom point of contact profile for a specific domain.

Purpose

The **manageItfimPointOfContact** command can perform the following operations on a point of contact profile when used with the appropriate parameters:

- list
- listCallbacks
- create (using a response file)
- create the response file
- view
- activate

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimPointOfContact {-operation operator
  -fimDomainName name [options]}
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-uuid ID
-signInCallbackIds callback1,callback2
-signOutCallbackIds callback1,callback2
-locaIdCallbackIds callback1,callback2
-authenticationCallbackIds callback1,callback2
-fileId output_file | input_file
```

The use of these parameters depends on the operator you chose to use.

Parameters

The following parameters are available for use with the **manageItfimPointOfContact** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform on the domain. Valid values are listed in the following table.

Table 31. Values for the manageItfimPointOfContact -operation parameter

Value	Description and requirements
view	View the properties of a point of contact profile and its callbacks. When you use this operator, you must also use the following parameters: **uuid** *ID* Unique identifier of the existing point of contact profile. You can determine the uuid of existing point of contact profiles by running the list operation, described previously.

Table 31. Values for the manageItfimPointOfContact -operation parameter (continued)

Value	Description and requirements
activate	Activate a specific point of contact profile. When you use this operator, you must also use the following parameters: **uuid** *ID* Unique identifier of the existing point of contact profile. You can determine the uuid of existing point of contact profiles by running the list operation, described previously. **Note:** If you are activating a WebSphere profile, the following default properties are used: SOAP Port=9444 Authorization type=Allow Authenticated users to access SOAP endpoints Authentication type=Basic If your environment requires different settings, you must pass in a text file that contains the appropriate settings. For information, refer to "Point of contact settings override" on page 277.
delete	Delete a specific custom point of contact profile. **Note:** You cannot delete a default point of contact profile. You can only delete point of contacts that you have created. The default point of contact profiles are set to Read Only to prevent accidental deletion. When you use this operator, you must also use the following parameters: **uuid** *ID* Unique identifier of the existing point of contact profile. You can determine the uuid of existing point of contact profiles by running the list operation, described previously.
list	List all of the existing point of contact profiles for a specific domain.
listCallbacks	List the enabled callbacks in a domain.

Table 31. Values for the manageItfimPointOfContact -operation parameter (continued)

Value	Description and requirements
createResponseFile	Create a response file that you will use to create a point of contact profile. You have the option to create a response file for a new point of contact profile or create a response file that is based on an existing point of contact profile. **New point of contact profile:** When you use this operator to create response file for a new point of contact, you must also specify the following parameters: **signInCallbackIds** *callback1,callback2* **signOutCallbackIds** *callback1,callback2* **localIdCallbackIds** *callback1,callback2* **authenticationCallbackIds** *callback1,callback2* **fileId** *output_file* Specify the file name and path for the response file that is created by this command. **Based on existing point of contact profile:** When you use this operator to create a response file that is based on an existing point of contact profile, you must also specify the following parameters: **uuid** *ID* Unique identifier of the existing point of contact profile. You can determine the uuid of existing point of contact profiles by running the list operation, described previously. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. After you have created the response file, open it with a text editor, review the attributes that are defined in the file, make changes as required by your environment, and then save and close the file. For information about the content of the response file, see "Point of contact response file" on page 274.
create	Create a point of contact profile using a response file. When you use this operator, you must also use the following parameters: **fileId** *input_name* This parameter specifies the name and path of the response file that you are using as input. You can create the response file using the createResponseFile operator.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name can be a string with characters of any type.

-uuid *ID*
An identifier string that uniquely identifies the resource you want to operate on.

-signInCallbackIds *callback*
A comma-separated list of callbacks used by the point of contact for sign-in actions.

-signOutCallbackIds *callback*
A comma-separated list of callbacks used by the point of contact for sign-out actions.

-localIdCallbackIds *callback*
A comma-separated list of callbacks used by the point of contact for the local ID.

-authenticationCallbackIds *callback*
A comma-separated list of callbacks used by the point of contact for sign authentication.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file or creating a point of contact profile. The value used with this parameter is the file name and path of a response file that is read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

View point of contact details:

```
$AdminTask manageItfimPointOfContact {-operation view
  -fimDomainName domain1
  -uuid uuid8f3d17a-0107-w712-q35b-b0c5ecc605ba}
```

Activate a point of contact:

```
$AdminTask manageItfimPointOfContact {-operation activate
  -fimDomainName domain1
  -uuid uuid8f3d17a-0107-w712-q35b-b0c5ecc605ba}
```

Delete a custom point of contact profile:

```
$AdminTask manageItfimPointOfContact {-operation delete
  -fimDomainName domain1
  -uuid uuid3e8de4e8-0119-1a2d-9443-c4944d126cc1}
```

List all the point of contact profiles that are defined in a domain:

```
$AdminTask manageItfimPointOfContact {-operation list
  -fimDomainName domain1}
```

List all the callbacks that are enabled in the domain:

```
$AdminTask manageItfimPointOfContact {-operation listCallbacks
  -fimDomainName domain1}
```

Create a response file to create a point of contact profile:

```
$AdminTask manageItfimPointOfContact {-operation createResponseFile
  -fimDomainName domain1
  -signInCallbackIds genericPocSignInCallback,wasPocSignInCallback
  -signOutCallbackIds genericPocSignOutCallback
  -localIdCallbackIds genericPocLocalIdentityCallback
  -authenticationCallbackIds genericPocAuthenticateCallback
  -fileId c:\home\files\temp\empty.xml}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a point of contact profile. After you have created this file, open it with a text editor and define the attributes in it so that they are correct for your environment.

Create a response file based on an existing point of contact profile:

```
$AdminTask manageItfimPointOfContact {-operation createResponseFile
  -fimDomainName domain1
  -uuid uuid8f3d17a-0107-w712-q35b-b0c5ecc605ba
  -fileId c:\home\files\temp\empty.xml}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a point of contact profile or modifying point of contact properties. After you have created this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a point of contact profile:

```
$AdminTask manageItfimPointOfContact {-operation create
  -fimDomainName domain1
  -fileId c:\home\files\temp\empty.xml}
```

Note: The file specified here is the response file and is used as input. Before running this command, you must have opened the response file with a text editor and ensured that the attributes that are defined in the file are correct for your environment.

Point of contact response file

Before you can create a point of contact profile using the **manageItfimPointOfContact** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

You can create a response file for creating a new partner by running the following command:

New point of contact profile

```
$AdminTask manageItfimPointOfContact {-operation createResponseFile
  -fimDomainName name
  -uuid ID
  -fileId filename}
```

Existing point of contact profile

Create a response file for creating a point of contact that is based on an existing point of contact by running the following command:

```
$AdminTask manageItfimPointOfContact {-operation createResponseFile
  -fimDomainName name
  -signInCallbackIds callback,callback
  -signOutCallbackIds callback
  -localIdCallbackIds callback
  -authenticationCallbackIds callback
  -fileId filename}
```

After you have run either of these commands, a response file is created. The content of the file differs depending on the properties specified in the command or in the existing point of contact.

Note: You must follow these steps to ensure that any custom properties that are used by your callbacks are included:

1. Open the response file with a text editor.
2. Review the attributes that are defined in the file.
3. Specify the type of federation that you want to create.
4. Save and close the file.

open the response file with a text editor, review the attributes that are defined in the file, make changes to specify the appropriate values, and then save and close the file, before you use it to run the create operation.

Examples of the response file is in the following directories:

AIX, Linux or Solaris

```
/opt/IBM/FIM/examples/responsefiles
```

Windows

```
C:\Program Files\IBM\FIM\examples\responsefiles
```

Parameters

The following descriptions show the types of parameters that are used in the response files. However, the actual parameters used in your XML response file depend on your environment and the callbacks you are using. For an example of a response file, see "Examples" on page 276.

Table 32. Parameters used in point of contact response files

Parameter	Value	Description
profileName=	*name*	Name of the point of contact profile.
Description=	*text*	Description of the profile.
signIn.*INDEX*=	*callbackID*	One or more callback IDs that are used for signing in. The INDEX represents the order in which this callback is invoked in the signing chain. It starts at 1. The callback ID identifies the callback module that is invoked.
CALLBACKID.PROPERTYNAME=)	*value*	Specifies a callback module and indicates that a property is used with it.
signOut.*INDEX*=	*callbackID*,*callbackID*,	One or more callback IDs that are used for signing out. The INDEX represents the order in which this callback is invoked in the signing chain. It starts at 1. The callback ID identifies the callback module that is invoked.
CALLBACKID.PROPERTYNAME=)	*value*	Specifies a callback module and indicates that a property is used with it.
localId.*INDEX*=	*callbackID*,*callbackID*,	One or more callback IDs that are used for the local Id. The INDEX represents the order in which this callback is invoked in the signing chain. It starts at 1. The callback ID identifies the callback module that is invoked.
CALLBACKID.PROPERTYNAME=)	*value*	Specifies a callback module and indicates that a property is used with it.

Table 32. Parameters used in point of contact response files (continued)

Parameter	Value	Description
authentication.*INDEX*=	*callbackID,callbackID,*	One or more callback IDs that are used for the authentication. The INDEX represents the order in which this callback is invoked in the signing chain. It starts at 1. The callback ID identifies the callback module that is invoked.
***CALLBACKID.PROPERTYNAME*=**	*value*	Specifies a callback module and indicates that a property is used with it.

Examples

Command example: The following example shows how to use the create command and specify the response file:

```
$AdminTask manageItfimPointOfContact {-operation create -fimDomainName domain1
  -fileId c:\home\files\temp\POCprops.xml}
```

Response file example: The following example describes a response file that can be used with the activate operation.

As you review the example, remember that a point of contact profile has a hierarchy in which there are:

```
0 or 4 callback types
  each callback type has 1 or more ordered callbacks
   each callback has 0 or more arbitrary properties
```

For example:

```
signIn.INDEX=CALLBACKID
  CALLBACKID.PROPERTYNAME1=value
  CALLBACKID.PROPERTYNAME2=value
```

Note: INDEX is a number that represents the order of the callback ID for that particular type (`signIn` in the example). Then, the callback ID can have properties added to it by prefixing the property name with the callback ID. The command line interface decomposes the response and add the properties to the callback and assign them to the proper type in the given order. The response file does not look as a key=value pair in the XML but it is effectively the same.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0" class="java.beans.XMLDecoder">
 <object class="java.util.HashMap">
  <void method="put">
   <string>signIn.1</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>wasPocSignInCallback</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>wasPocAuthenticateCallback.authentication.macros</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>%FEDID%</string>
```

```
    </void>
    <void method="add">
     <string>%FEDNAME%</string>
    </void>
    <void method="add">
     <string>%PARTNERID%</string>
    </void>
    <void method="add">
     <string>%ACSURL%</string>
    </void>
    <void method="add">
     <string>%SSOREQUEST%</string>
    </void>
    <void method="add">
     <string>%TARGET%</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>profileName</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>testwaspoc</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>profileDescription</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>WebSphere Point of Contact Profile</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>localId.1</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>wasPocLocalIdentityCallback</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>signOut.1</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>wasPocSignOutCallback</string>
    </void>
   </object>
  </void>
  <void method="put">
   <string>authentication.1</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>wasPocAuthenticateCallback</string>
    </void>
   </object>
  </void>
 </object>
</java>
```

Point of contact settings override

Use default properties if you are activating a WebSphere profile. If you need values other than the default values, you must create a settings file and use it with the activate command to override those values.

Purpose

When you use the activate operation of the manageItfimPointOfContact command to activate a WebSphere Application Server point of contact, the default values are used:

SOAP Port=9444

Authorization type=Allow authenticated users to access SOAP endpoints

Authentication type=Basic

If your environment requires different settings, you must pass in a text file that contains the appropriate settings.

Content of text file

Create a text file with one or more of the key-value pairs you want to override:

```
SOAP.PORT=port_number
SOAP.AUTHORIZATION=UNAUTHENTICATED|AUTHENTICATED|GROUP
SOAP.GROUPNAME=group_name
SOAP.AUTHENTICATION=BASIC|CLIENTCERTIFICATE
```

Note: You must include SOAP.GROUPNAME=*group_name* if you use SOAP.AUTHORIZATION=GROUP.

Save the file and take note of its location, such as `/home/files/temp/WASPOCSettings.txt`. Then, run the activate operation as described in the following sections.

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimPointOfContact {-operation activate
  -fimDomainName name
  -uuid ID
  -fileId input_file}
```

Parameters

The following parameters must be used to activate the WebSphere Application Server point of contact server with custom settings.

-operation activate
: Required parameter and value for activating point of contact server.

-fimDomainName *name*
: Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name can be a string with characters of any type.

-uuid *ID*
: An identifier string that uniquely identifies the resource you want to operate on. To determine the ID of the enabled point of contact servers in a specific domain, run the manageItfimPointOfContact command with the list operation as described in “manageItfimPointOfContact” on page 270.

-fileId *input_file*
: The text file that contains the key-value pairs that you want to set for the WebSphere Application Server point of contact that you are activating.

Examples

The following example shows an example of the correct syntax:

Activate a point of contact with custom settings:

```
$AdminTask manageItfimPointOfContact {-operation activate
  -fimDomainName domain1
  -uuid uuid8f3d17a-0107-w712-q35b-b0c5ecc605ba
  -fileId c:\home\files\temp\WASPOCSettings.txt}
```

manageItfimKeys

Use this command to manage keys and keystores in your IBM Tivoli Federated Identity Manager environment.

Purpose

This command, when used with the appropriate parameters, can perform the following operations on the keys and keystores in your environment:

- list
- delete
- enable
- disable
- import
- export

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimKeys {-operation operator -fimDomainName name
  [optional_parameters]}
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-keystorePassword password
-keystoreName name
-trustedKeystore true | false
-keyAlias alias
-keystoreFormat  JKS | PKCS12 | PEM
-fileId output_file | input_file
-sourceKeystorePwd password
-exportPrivateKey true | false
```

Parameters

The following parameters are available for use with the **manageItfimKey** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform on the domain. Valid values are listed in the following table.

Table 33. Values for the manageItfimKey -operation parameter

Value	Description and requirements
list	List all of the existing keystores in a domain or keys in a keystore. When you use this operator, you must also use the following parameters, ***if you are listing keys in a keystore***: **keystoreName** *name* Specify the name of the keystore. **keystorePassword** *password* Specify the password of the keystore.
delete	Delete a key or keystore. When you use this operator, you must also use the following parameters: **keystoreName** *name* Specify the name of the keystore. **keystorePassword** *password* Specify the password of the keystore. This parameter is required *only if you are deleting a key from a keystore.* **keyAlias** *name* Specify the alias of the key. This parameter is required *only if you are deleting a key from a keystore.*
enable	Enable a key. When you use this operator, you must also use the following parameters: **keystoreName** *name* Specify the name of the keystore where the key is located. **keystorePassword** *password* Specify the password of the keystore. **keyAlias** *name* Specify the alias of the key.
disable	Disable a key. When you use this operator, you must also use the following parameters: **keystoreName** *name* Specify the name of the keystore where the key is located. **keystorePassword** *password* Specify the password of the keystore. **keyAlias** *name* Specify the alias of the key.

Table 33. Values for the manageItfimKey -operation parameter (continued)

Value	Description and requirements
import	Import a key or keystore. When you use this operator, you must also use the following parameters. **Importing a keystore:** **keystoreName** *name* Specify the name that you want to give to the keystore that you are importing. **keystorePassword** *password* Specify the password of the keystore. It retains this password after it is imported. **trustedKeystore** `true	false` Specify whether the keystore is a keystore (false) or a truststore (true). This parameter is required only if you are importing a keystore.  **fileId** *input_file* Specify the file name and path of the file that you are importing.  **keystoreFormat** `JKS	PEM| PKCS12` Specify the file format used by the original keystore in which the key was stored. **Importing a key:** **keystoreName** *name* Specify the name of the keystore to which you are importing the key. **keystorePassword** *password* Specify the password of the keystore. **keyAlias** *name* Specify the alias of the key that is in the source file and that is used in the new keystore. **keystoreFormat** `JKS	PEM| PKCS12` Specify the file format used by the original keystore in which the key was stored. **fileId** *input_file* Specify the file name and path of the file that you are importing. **sourceKeystorePwd** *password* The password of the keystore from which the key originated.

Table 33. Values for the manageItfimKey -operation parameter (continued)

Value	Description and requirements
export	Export a key. When you use this operator, you must also use the following parameters: **keystoreName** *name* Specify the name of the keystore. **keystorePassword** *password* Specify the password of the keystore. **keyAlias** *name* Specify the alias of the key. **keystoreFormat JKS | PEM| PKCS12** Specify the format of the keystore that contains the exported key. **fileId** *output_file* Specify the file name and path of the keystore file that is created. **Attention:** A new keystore is created with this parameter. Specify a unique keystore file name. If you specify an existing keystore, it is overwritten. **exportPrivateKey true | false** Indicate whether you want to export both the private and public key. True indicates that the private key is exported with the public key. False indicates that only the public key is exported.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name must be a string with characters of any type.

-keystorePassword *password*
This parameter is required for all operations except when **list** is used to list all the keystores in the domain. The value used with this parameter is the password of the keystore. The name must be a string with characters of any type.

-keystoreName *name*
This parameter is required for all operations except when **list** is used to list all the keystores in a domain. The value used with this parameter is the name of the keystore. The name must be a string with characters of any type.

-trustedKeyStore *true* | *false*

This parameter is required if you are importing a keystore. The value used with this parameter indicates whether the keystore is a truststore, in which the keys of your partner, such as those you use for validation and encryption, and Certificate Authority certificates are stored, or if it is a keystore, in which your keys, such as those you use for signing and decryption are stored.

Use **true** to indicate that the keystore is a truststore. The default value is **false**. The values must be in lowercase.

-keyAlias *alias*
This parameter is required if you are deleting a key, enabling a key, disabling a

key, importing a key, or exporting a key. The value used with this parameter is the alias of the key on which the operation occurs. The name must be a string with characters of any type.

-keystoreFormat *JKS | PKCS12 | PEM*
This parameter is required if you are importing or exporting a key. The value used with this parameter identifies the file format of the keystore that contains the key. The values must be in uppercase.

-fileId *output_file | input_file*
This parameter is required if you are importing a keystore, importing a key, or exporting a key. The value used with this parameter is the file name and path of the key or keystore that is being imported (input file) or the key that is being exported (output file). The path and file name must be valid for the operating system being used.

-sourceKeystorePwd *password*
This parameter is required if you are importing a keystore. The value used with this parameter is the password of the keystore from which the key originated.

-exportPrivateKey *true | false*
This parameter is required if you are exporting a key. The value used with this parameter indicates whether the private key should be exported with the public key. Use **true** to indicate that the both keys should be exported. Use **false** to indicate that only the public key should be exported. The default value is **false**. The values must be in lowercase.

Attention: Do not include your private key if you are exporting a key to provide to your partner.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the keystores in a domain:
```
$AdminTask manageItfimKeys {-operation list -fimDomainName domain1}
```

List all the keys in a keystore:
```
$AdminTask manageItfimKeys {-operation list -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly}
```

Delete a keystore:
```
$AdminTask manageItfimKeys {-operation delete -fimDomainName domain1
  -keystoreName DefaultKeyStore}
```

Delete a key:
```
$AdminTask manageItfimKeys {-operation delete -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly -keyAlias spkey}
```

Enable a key:
```
$AdminTask manageItfimKeys {-operation enable -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly -keyAlias spkey}
```

Disable a key:
```
$AdminTask manageItfimKeys {-operation disable -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly -keyAlias spkey}
```

Import a keystore:
```
$AdminTask manageItfimKeys {-operation import -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly
  -trustedKeystore false -fileId c:\temp\newkey.jks}
```

Import a key:

```
$AdminTask manageItfimKeys {-operation import -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly
  -keyAlias spkey -keystoreFormat JKS
  -fileId c:\temp\newkey.jks -sourceKeystorePwd testonly}
```

Export a key:

```
$AdminTask manageItfimKeys {-operation export -fimDomainName domain1
  -keystoreName DefaultKeyStore -keystorePassword testonly
  -keyAlias spkey -keystoreFormat JKS
  -fileId c:\temp\newkey.jks -exportPrivateKey false}
```

manageItfimNameIdSvc

Use this command to manage the alias service in your IBM Tivoli Federated Identity Manager environment.

Purpose

This command, when used with the appropriate parameters, can perform the following operations on the alias service in your environment:

- view
- configure
- modify
- add a host
- remove a host
- modify a host

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimNameIdSvc {-operation operator -fimDomainName name
  [optional_parameters]}
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-configurationType ldap | jdbc
-userSearchFilter uid=@USERID@
-userIdAttribute uid
-baseDn dc=com
-rootSuffix cn=itfim
-keystoreName name
-hostName hostname
-hostPort port
-bindDn name
-bindPassword password
-minConnections number_of_connections
-maxConnections number_of_connections
-hostOrder sequence_number
-mode ro | rw | wo
-uuid unique_server_identifier
```

Parameters

The following parameters are available for use with the **manageItfimNameIdSvc** command:

-operation *operator*

Required parameter. The value used with this parameter specifies the operation to perform on the domain. Valid values are listed in the following table.

Table 34. Values for the manageItfimNameIdSvc -operation parameter

Value	Description and requirements
view	Show the configuration of the service.
configure	Configure the Name ID service for a specific provider type. When you use this operator, you must also use the following parameters. **To configure the service to use a JDBC provider:** **configurationType** `jdbc` **To configure the service to use an LDAP provider without SSL:** **configurationType** `ldap` **userSearchFilter** *uid=@USERID@* **userIdAttribute** *uid* **baseDn** *dc=com* **rootSuffix** *cn=itfim* **To configure the service to use an LDAP provider with SSL enabled:** **configurationType** `ldap` **userSearchFilter** *uid=@USERID@* **userIdAttribute** *uid* **baseDn** *dc=com* **rootSuffix** *cn=itfim* **keystoreName** *name*

modify	Modify the properties of the name identifier service. You might want to use the **view** operator before running **modify** so that you know what the existing properties are. When you use this operator, you must also use any other parameter that is already configured.
	
	For example, if you want to change the current baseDn and rootSuffix, you would run the following command:
	
	`manageItfimNameIdSvc {-operation modify`
	`-fimDomainName domain1`
	`-baseDn dc=org -rootSuffix cn=itfim2}`
	
	You can also change multiple properties at the same time by specifying them in the same command string.

Table 34. Values for the manageItfimNameIdSvc -operation parameter (continued)

Value	Description and requirements
addHost	Add an LDAP host to the queue. When you use this operator, you must also use the following parameters: **To add an LDAP server in a specific position:** **hostName** *name* **hostPort** *port* **bindDn** *name* **bindPassword** *password* **minConnections** *2* **maxConnections** *10* **mode ro | rw | wo** **hostOrder** *a number between 0 and the number of hosts in the queue -1* **To add an LDAP server at the last position in the queue:** **hostName** *name* **hostPort** *port* **bindDn** *name* **bindPassword** *password* **minConnections** *2* **maxConnections** *10* **mode ro | rw | wo**
removeHost	Remove a host from the queue. When you use this operator, you must also use this parameter: **uuid** *unique_server_identifier*
modifyHost	Modify a host in the queue. When you use this operator, you must also use the following parameters. **uuid** *unique_server_identifier* **(any other parameter that is already configured)**

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The name must be a string with characters of any type.

-configurationType *ldap | jdbc*
This parameter is required when you are configuring the alias service. The value used with this parameter is the type of database to use with the service. The values must be specified in lowercase.

-userSearchFilter *uid=@USERID@*
This parameter is required when you are configuring an LDAP database for the service. The value used with this parameter is the LDAP user search filter. The default value is uid=@USERID@.

-userIdAttribute *uid*
This parameter is required if you are configuring an LDAP database for the service. The value used with this parameter is the ID attribute for a user ID. The default value is uid.

-baseDn *dc=com*
This parameter is required when you are configuring an LDAP database for the service. The value used with this parameter is the base distinguished name. The default value is dc=com.

-rootSuffix *cn=itfim*
This parameter is required when you are configuring an LDAP database for the service. The value used with this parameter is the LDAP root suffix. The default value is cn=itfim.

-keystoreName *name*
This parameter is required when you are configuring an LDAP database that uses SSL. The value used with this parameter is the name of the IBM Tivoli Federated Identity Manager truststore. The name must be a string with characters of any type.

-hostName *hostname*
This parameter is required when you are adding, removing, or modifying a host. The value used with this parameter is the host name or IP address of the database you are adding, removing, or modifying.

Note: The format of the value specified is not validated and a verification connection is not attempted.

-hostPort *port*
This parameter is required when you are adding, removing, or modifying a host. The value used with this parameter is the listening port of the database you are adding, removing, or modifying.

Note: The command validates that the value is a number. It does not verify that the port is active.

-bindDn *name*
This parameter is required when you are adding or modifying a host. The value used with this parameter is the distinguished name that is used to bind to the LDAP server.

-bindPassword *password*
This parameter is required when you are adding or modifying a host. The value used with this parameter is the password that is required to bind to the LDAP server.

-minConnections *number*
This parameter is required when you are adding a host. It can be modified when you modify a host. The value used with this parameter is the minimum number of connections that can be made to the LDAP server. The value must be an integer of 0 or more. The default value is 2.

-maxConnections *number*
This parameter is required when you are adding a host. It can be modified when you modify a host. The value used with this parameter is the maximum number of connections that can be made to the LDAP server. The value must be an integer of 0 or more and must be more than the value for minConnections. The default value is 10.

-mode *ro | rw | wo*
This parameter is required when you are adding a host. It can be modified when you modify a host. The value used with this parameter is the mode of the LDAP server. Valid values are rw (read/write), ro (read only), and wo (write only). The values must be specified in lowercase.

-hostOrder *sequence_number*

This parameter is the sequence number that defines the order in the server queue where the host is inserted. The number 0 is the first host. The order in the queue reflects the order in which servers are contacted.

If this parameter is not specified (or if a value less than 0, such as -1 is specified), the host is appended to the end of the queue. A value that is larger than or equal to the number of hosts in the queue also results in the host to be added to the end of the queue.

-uuid *unique_server_identifier*
An identifier string that uniquely identifies the server you want to operate on.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

Configure a JDBC configuration:

```
$AdminTask manageItfimNameIdSvc {-operation configure
  -fimDomainName domain1 -configurationType jdbc}
```

Configure LDAP without SSL:

```
$AdminTask manageItfimNameIdSvc {-operation configure
  -fimDomainName domain1 -configurationType ldap
  -userSearchFilter uid=@USERID@ -userIdAttribute uid
  -baseDn dc=com -rootSuffix cn=itfim }
```

Configure LDAP with SSL

```
$AdminTask manageItfimNameIdSvc {-operation configure
  -fimDomainName domain1 -configurationType ldap
  -userSearchFilter uid=@USERID@ -userIdAttribute uid
  -baseDn dc=com -rootSuffix cn=itfim
  -keystoreName DefaultKeyStore}
```

Add an LDAP server to the configuration to the end of the queue:

```
$AdminTask manageItfimNameIdSvc {-operation addHost
  -fimDomainName domain1 -hostName host1
  -hostPort port1 -bindDn defaultdn -bindPassword passw0rd
  -minConnections 2 -maxConnections 10 -mode rw}
```

Add an LDAP server to the configuration in a specific position:

```
$AdminTask manageItfimNameIdSvc {-operation addHost
  -fimDomainName domain1 -hostName host1
  -hostPort port1 -bindDn cn=root -bindPassword passw0rd
  -minConnections 2 -maxConnections 10 -mode rw -hostOrder 3}
```

Remove an LDAP server from the queue:

```
$AdminTask manageItfimNameIdSvc {-operation removeHost
  -fimDomainName domain1
  -uuid uuid19f150510-0119-1036-8b18-c8e3700ca101 }
```

Modify an LDAP server in the queue:

Note: In the following example, all options that can be modified are included.

```
$AdminTask manageItfimNameIdSvc {-operation modifyHost
  -fimDomainName domain1 -hostName host1
  -uuid uuid19f150510-0119-1036-8b18-c8e3700ca101
  -hostPort port1 -bindDn cn=root -bindPassword passw0rd
  -minConnections 2 -maxConnections 10 -mode rw -hostOrder 3}
```

View the configuration of the service and the queue of hosts:

```
$AdminTask manageItfimNameIdSvc {-operation view
  -fimDomainName domain1}
```

Modify the properties of the service:

```
$AdminTask manageItfimNameIdSvc {-operation modify
  -fimDomainName domain1
  -baseDn dc=org -rootSuffix cn=itfim2}
```

Add a negative value for the hostOrder:

```
$AdminTask manageItfimNameIdSvc {-operation addHost
-fimDomainName domain1 -hostName host1
-hostPort port1 -bindDn cn=root -bindPassword passw0rd -minConnections 2
-maxConnections 10 -mode rw -hostOrder {-3}}
```

Note: For a negative value to be specified in the hostOrder parameter, it must be surrounded by braces otherwise a number format exception occurs.

manageItfimReports

Use the **manageItfimReports** command to manage IBM Tivoli Federated Identity Manager reports.

Purpose

The **manageItfimReports** can perform the following operations for viewing, running, and deleting reports when used with the appropriate parameters:

- listActive
- listArchived
- listRunnable
- create the response file
- delete
- run

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimReports {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-reportDesign name
-renderType pdf | html
-fileId output_file | input_file
```

The use of these parameters depends on the operator you specify.

Parameters

The following parameters are available for use with the **manageItfimReports** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 35. Values for the manageItfimReports -operation parameter

Value	Description and requirements
listActive	Use this operation to create a list of the reports that are being run during a session. The list is cleared when WebSphere Application Server is restarted.
listArchived	Use this operation to create a list of the archived reports. The list is cleared when WebSphere Application Server is restarted.
listRunnable	Use this operation to create a list of the reports that are available to run.
createResponseFile	Create a response file in which you specify what information to include in the report. When you use this operator, you must also use the following parameters: **reportDesign** *name* The following pre-defined reports are available: • administrative_events.rptdesign • sso_summary.rptdesign **fileId** *output_file* After creating the response file, open it with a text editor. Review the attributes that are defined in the file, and specify the type of report that you want to run. Then save and close the file. For information about the content of the response file, see "Administrative events report response file" on page 291 or "Single sign-on summary report response file" on page 292.
run	Run a IBM Tivoli Federated Identity Manager report. When you use this operator, you must also use the following parameters: **reportDesign** *name* The following pre-defined reports are available: • administrative_events.rptdesign • sso_summary.rptdesign **fileId** *output_file* **renderType** *pdf* | *html*
delete	Delete a report. When you use this operator, you must also specify the **reportDesign** *name* parameter and provide it with the name of the report to delete.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-reportDesign *name*
The name of the report design. There are two pre-defined reports available:

- administrative_events.rptdesign
- sso_summary.rptdesign

-renderType *pdf | html*
This parameter is required when you are running a report. The value used with this parameter is format of the report. Specify either pdf or html.

-fileId *output_file | input_file*
The value used with this parameter is the file name and path of a response file that is read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

Create a response file:

```
$AdminTask manageItfimReport {-operation createResponseFile
  -fimDomainName domain1 -reportDesign administrative_events.rptdesign
  -fileId c:\dnload\admin_eventsreport.txt}
```

After creating the response file, open it with a text editor and review the attributes that are defined in the file. Specify the type of report that you want to run, and then save and close the file.

Run a report:

```
$AdminTask manageItfimReport {-operation run -fimDomainName domain1
  -reportDesign administrative_events.rptdesign
  -fileId c:\dnload\admin_eventsreport.txt -renderType pdf}
```

Note: Before you can run a report, you must first create the response file.

Delete a report:

```
$AdminTask manageItfimReports {-operation delete -fimDomainName domain1
  -reportDesign sso_summary_events_042610182942.pdf}
```

List Runnable reports:

```
$AdminTask manageItfimReports {-operation listRunnable
  -fimDomainName domain1}
```

List Active reports

```
$AdminTask manageItfimReports {-operation listActive
  -fimDomainName domain1}
```

List Archived Reports

```
$AdminTask manageItfimReports {-operation listArchived
  -fimDomainName domain1}
```

Administrative events report response file

Before you can run a report using the **run operation with the manageItfimReports** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

Create a response file for running an administrative events report by running the following command:

```
$AdminTask manageItfimReports {-operation createResponseFile -fimDomainName name
   -reportDesign administrative_events.rptdesign
  -fileId file}
```

After you have run this command, a response file is created as follows:

```
#Response File generated for administrative_events.rptdesign
#Wed Nov 28 00:14:52 CST 2007
Administrator=Enter a value here
Date_Range_Start=Enter a value here
Event_Type=Enter a value here
Date_Range_End=Enter a value here
```

Open the file with a text editor, review the attributes that are defined in the file, make changes to specify the type of report you want to run, and then save and close the file.

Parameters

Administrator=*Enter a value here*
The value can be `All User IDs` (case sensitive and specified as shown here), or a specific administrator user, such as `fimadmin`.

Date_Range_Start=*Enter a value here*
Specify the date and time on which the report should begin. The date must be specified in mm/dd/year format. The time must be specified in 12-hour format with hour, minute, seconds (hh:mm:ss) and the abbreviation AM or PM in uppercase, for example 12:00:00 AM.

Event_Type=*Enter a value here*
Specify the events to include in the report. Valid values are `All Events`, `Federation Events`, `Federation Partner Events`, `Web Service Partner Events`, or `Audit Configuration Events`.

Date_Range_End=*Enter a value here*
Specify the date and time on which the report should end. The date must be specified in mm/dd/year format. The time must be specified in 12-hour format with hour, minute, seconds (hh:mm:ss) and the abbreviation AM or PM in uppercase, for example 12:00:00 AM.

Example

Following is a report response file that has been edited. Values that are appropriate to a specific environment have been added.

```
 #Response File generated for administrative_events.rptdesign
#Wed Nov 28 00:14:52 CST 2007
Administrator=All User IDs
Date_Range_Start=01/01/2007 12:00:00 AM
Event_Type=All Events
Date_Range_End=01/31/2007 12:00:00 AM
```

Single sign-on summary report response file

Before you can run a report using the **run operation with the manageItfimReports** command, you must create a response file, and then edit the response file so that it contains the appropriate values for your environment.

Create a response file for running a single sign-on summary report by running the following command:

```
$AdminTask manageItfimReports {-operation createResponseFile -fimDomainName name
   -reportDesign sso_summary.rptdesign
   -fileId file}
```

After you have run this command, a response file is created as follows:

```
#Response File generated for sso_summary.rptdesign
#Wed Nov 28 00:14:52 CST 2007
Date_Range_Start=Enter a value here
Event_Type=Enter a value here
Date_Range_End=Enter a value here
User=All User IDs
```

Open the file with a text editor, review the attributes that are defined in the file, make changes to specify the type of report you want to run, and then save and close the file.

Parameters

Date_Range_Start=*Enter a value here*
Specify the date and time on which the report should begin. The date must be specified in mm/dd/year format. The time must be specified in 12-hour format with hour, minute, seconds (hh:mm:ss) and the abbreviation AM or PM in uppercase, for example 12:00:00 AM.

Event_Type=*Enter a value here*
Specify the events to include in the report. Valid values are All Events, Successful Single Sign-On Events, or Failed Single Sign-On Events.

Date_Range_End=*Enter a value here*
Specify the date and time on which the report should end. The date must be specified in mm/dd/year format. The time must be specified in 12-hour format with hour, minute, seconds (hh:mm:ss) and the abbreviation AM or PM in uppercase, for example 12:00:00 AM.

User=*Enter a value here*
Specify a specific user name. Valid values are `All User IDs` or an individual user name.

Example

Following is a report response file that has been edited. Values that are appropriate to a specific environment have been added.

```
 #Response File generated for sso_summary.rptdesign
#Wed Nov 28 00:14:52 CST 2007
Date_Range_Start=01/01/2007 12:00:00 AM
Event_Type=All Events
Date_Range_End=01/31/2007 12:00:00 AM
User=elaine
```

reloadItfimManagementService

Use the **reloadItfimManagementService** command to reload the IBM Tivoli Federated Identity Manager management service. Reloading the management service is necessary if you have added new plug-ins to your environment.

Purpose

The **reloadItfimManagementService** reloads the IBM Tivoli Federated Identity Manager management service.

Syntax

The command syntax is as follows:

```
$AdminTask reloadItfimManagementService
```

No parameters are used with this command.

Examples

```
$AdminTask reloadItfimManagementService
```

reloadItfimRuntime

Use the **reloadItfimRuntime** command to reload the IBM Tivoli Federated Identity Manager runtime. Reloading the runtime is necessary if you have made changes to the system configuration.

Purpose

The **reloadItfimRuntime** reloads the IBM Tivoli Federated Identity Manager runtime.

Syntax

The command syntax is as follows:

```
$AdminTask reloadItfimRuntime {-fimDomainName name [optional_parameters]}
```

where the `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-serverName server
-nodeName node
-cellName cell
```

Parameters

The following parameters are used with the **reloadItfimRuntime** command:

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is performed. The value must be a string for an existing domain name.

-serverName *server*
Optional parameter. The value used with this parameter is the name of the WebSphere Application Server on which the domain is running. The value must be a string for an existing server.

-nodeName *node*
Optional parameter. The value used with this parameter is the name of the WebSphere Application Server node on which the domain is running. The value must be a string for an existing node.

-cellName *cell*
Optional parameter. The value used with this parameter is the name of the WebSphere Application Server cell on which the domain is running. The value must be a string for an existing cell.

Note: If all of the optional parameters are left blank, all of the nodes in the specified domain are reloaded.

Examples

All parameters must be specified with this command.

```
$AdminTask reloadItfimRuntime {-fimDomainName domain1 -serverName server1
  -nodeName node1 -cellName cell1}
```

logoutItfimSaml20User

Use the **logoutItfimSaml20User** command to logout a user from a SAML 2.0 federation.

Purpose

The **logoutItfimSaml20User** command logs out the user of a SAML 2.0 single sign-on session. The action taken by the command depends on whether it is issued by an identity provider or a service provider.

Identity provider

When the command is invoked at an identity provider, then an attempt is made to logout the user from all service providers for which the identity provider issues assertions. The logout requests are sent to each service provider using the SOAP binding.

When a response is received from each service provider, the identity provider logs out the user locally, only if WebSEAL is being used as the point of contact.

Service provider

When the command is invoked at a service provider, the user is logged out locally, only if WebSEAL is being used as the point of contact. A service provider cannot log out a user at an identity provider.

Syntax

The command syntax is as follows:

```
$AdminTask logoutItfimSaml20User {-fimDomainName name -federationName name
  -userName name}
```

All of the parameters are required.

Parameters

The following parameters must be used with the **logoutItfimSaml20User** command:

-fimDomainName *name*

This parameter is required. The value used with this parameter is the name of the domain on which the operation is performed. The name must be a string with characters of any type.

-federationName *name*

Required parameter. The value used with this parameter is the name of the federation that the user is a member of. The value must be a string for an existing federation.

-userName *user*

Required parameter. The value used with this parameter is the name of the user who issued the single sign-on request.

Example

All parameters must be specified with this command.

```
$AdminTask logoutItfimSaml20User {-fimDomainName name -federationName name
  -userName name}
```

defederateItfimSaml20User

Use the **defederateItfimSaml20User** command to defederate a user from a SAML 2.0 federation.

Purpose

The **defederateItfimSaml20User** command defederates a user from a SAML 2.0 federation.

Syntax

The command syntax is as follows:

```
$AdminTask defederateItfimSaml20User {-fimDomainName name -federationName name
  -partnerId ID -userName name}
```

All of the parameters are required.

Parameters

The following parameters must be used with the **defederateItfimSaml20User** command:

-fimDomainName *name*
This parameter is required. The value used with this parameter is the name of the domain on which the operation is performed. The name must be a string with characters of any type.

-federationName *name*
Required parameter. The value used with this parameter is the name of the federation that the user is a member of. The value must be a string for an existing federation.

-partnerId *ID*
Required parameter. The value used with this parameter is the ID (also referred to as the protocol ID) of the partner from which the user should be defederated.

-userName *user*
Required parameter. The value used with this parameter is the name of the user who issued the single sign-on request.

Example

All parameters must be specified with this command.

```
$AdminTask defederateItfimSaml20User {-fimDomainName name -federationName name
  -partnerId https://idp1/FIM/sps/saml20IpFed/saml20 -userName name}
```

manageItfimSamlArtifactService

Use the **manageItfimSamlArtifactService** command to manage the SAML Artifact Service ofIBM Tivoli Federated Identity Manager.

Purpose

The **manageItfimSamlArtifactService** command can perform the following operations to manage the SAML 1.x Artifact Service when used with the appropriate parameters:

- list
- configure
- unconfigure

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimSamlArtifactService {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-configurationId unique_config_ID
-urlPathToSps URL
-artifactCacheLifetime number_of_seconds
-samlVersion 1.0|1.1
-sourceId ID
-artifactServiceId unique_service_ID
```

The use of these parameters depends on the operator you choose.

Parameters

The following parameters are available for use with the **manageItfimSamlArtifactService** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 36. Values for the manageItfimSamlArtifactService -operation parameter

Value	Description and requirements
list	Lists all of the configured SAML 1.x artifact services on a domain.

Table 36. Values for the manageItfimSamlArtifactService -operation parameter (continued)

Value	Description and requirements
configure	Configures a SAML 1.x artifact service with a WS-Trust endpoint and a SAML 1.x resolution endpoint. You can use the following parameters for this operation: **configurationId** *unique_ID* Required parameter. Specify the name of the configuration, which is also known as the federation name. The value must be a string for an existing configuration. **urlPathToSps** *URL* Required parameter. Specify the URL of the SPS service, which is typically `https://<hostname>/sps`. The value must be a string for a valid URL. **artifactCacheLifetime** *number_of_seconds* Required parameter. Specify the number of seconds for the lifetime of the artifact cache. **samlVersion** *1.0|1.1* Specify the SAML version of the artifact service. Valid values are 1.0 or 1.1. **sourceId** *ID* Optional parameter. Specify the SAML source ID. By default, this ID is generated as the Base64-encoded SHA-1 hash of the following string concatenation: `[urlPathToSps]/[configurationId]/saml[samlVersion]`
unconfigure	Unconfigures a SAML 1.x artifact service on a specific domain. The following parameter must be used with this operation: **artifactServiceId** *unique_ID* Specify the ID of the existing artifact service. The value must be a string for an existing artifact service.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-configurationId *unique_config_ID*
This parameter is required for the configure operation. The value used with this parameter is the name of the configuration, which is also known as the federation name. The value must be a string for an existing configuration.

-urlPathToSps *URL*
This parameter is required with the configure operation. The value used with this parameter is the URL of the SPS service, which is typically `https://<hostname>/sps`. The value must be a string for a valid URL.

-artifactCacheLifetime *number_of_seconds*
This parameter is required with the configure operation. The value used with this parameter is the number of seconds for the lifetime of the artifact cache. The value must be a positive integer ranging from 0 to 2,147,483,647. The default value is 300.

-samlVersion *1.0|1.1*
The SAML version of the artifact service. Valid values are 1.0 or 1.1. Used with the configure operation.

-sourceID *ID*
The SAML source ID. By default, this ID is generated as the Base64-encoded SHA-1 hash of the following string concatenation: `urlPathToSps/configurationId/samlsamlVersion`

-artifactServiceId
This parameter can be optionally used with the configure operation. The SAML source ID. By default, this ID is generated as the Base64-encoded SHA-1 hash of the following string concatenation: `[urlPathToSps]/[configurationId]/saml[samlVersion]`

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the configured SAML artifact services in a domain:

```
$AdminTask manageItfimSamlArtifactService {-operation list -fimDomainName domain1}
```

Configure a SAML1.x artifact service:

```
$AdminTask manageSamlArtifactService {-operation configure -fimDomainName localhost
   -server1 -configurationId artifactsvc -urlPathToSps https://localhost:9443/sps
   -artifactCacheLifetime 300 -samlVersion 1.1}
```

Unconfigure a SAML1.x artifact service:

```
$AdminTask manageItfimSamlArtifactService {-operation unconfigure -fimDomainName domain1
   -artifactServiceId service1}
```

manageItfimStsModuleType

Use the **manageItfimStsModuleType** command to manage an IBM Tivoli Federated Identity Manager Security Trust Service (STS) module type.

Purpose

The **manageItfimStsModuleType** command, when used with the appropriate parameters, can perform the following operations to manage token module types:

- list
- view

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimSTSModuleType {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameter is:

`-uuid` *module_type_uuid*

The use of these parameters depends on the operator you specify.

Parameters

The following parameters are available for use with the **manageItfimStsModuleType** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 37. Values for the manageItfimModuleType -operation parameter

Value	Description and requirements
list	List all of the existing STS module types.
view	View the details of a module type. When you use this operator, you must also specify the -uuid *module_type_uuid*

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-uuid *name*
An identifier string that uniquely identifies the resource on which to operate. This string is required for the view operation. To view the uuid of a Module Type, use the list operation.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the module types in a domain:

```
$AdminTask manageItfimStsModuleType {-operation list -fimDomainName domain1}
```

View module type details:

```
$AdminTask manageItfimStsModuleType {-operation view -fimDomainName domain1
  -uuid uuid1}
```

manageItfimStsModuleInstance

Use the **manageItfimStsModuleInstance** command to manage an IBM Tivoli Federated Identity Manager Security Trust Service (STS) module instance.

Purpose

The **manageItfimStsModuleInstance** command can perform the following operations to manage token module instances when used with the appropriate parameters:

- list
- view
- delete
- create (using a response file)
- create the response file
- modify

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimStsModuleInstance {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-uuid module_instance_uuid
-moduleId module_type_uuid
-fileId output_file | input_file
```

The use of these parameters depends on the operator you specify.

Parameters

The following parameters are available for use with the **manageItfimStsModuleInstance** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 38. Values for the manageItfimStsModuleInstance -operation parameter

Value	Description and requirements
list	List all of the existing STS module instance.
view	View the details of an STS module instance. When you use this operator, you must also specify the **uuid** *module_instance_name* parameter.

Table 38. Values for the manageItfimStsModuleInstance -operation parameter (continued)

Value	Description and requirements
createResponseFile	Create a response file used for creating an STS module instance. You have the option to create a response file for a new module instance. You can also create a response file that is based on an existing module instance. **New STS module instance:** When you use this operator to create response file for a new module instance, you must also specify the following parameters: **moduleId** *module_type_uuid* Specify the module type based on which the module instance is to be created. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. **Based on existing module instance:** When you use this operator to create a response file that is based on an existing module instance, you must also specify the following parameters: **uuid** *module_instance_uuid* Specify the uuid of the module instance name if you are creating a response file that is based on an existing module instance. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. After you have created the response file, open it with a text editor. Review the attributes that are defined in the file, and change them as required by your environment. Then save and close the file.
create	Create a module instance using a response file. When you use this operator, you must also specify the following parameters: **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.
delete	Delete a module instance. When you use this operator, you must also specify the **uuid** *module_instance_uuid* parameter.
modify	Modify a module instance using a response file. When you use this operator, you must also specify the following parameters: **uuid** *module_instance_uuid* Specify the name of the module instance that you want to modify. **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the createResponseFile parameter.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-uuid *module_type_uuid*
An identifier string that uniquely identifies the resource on which to operate.

-moduleId *module_type_uuid*
An identifier string that uniquely identifies a token module type.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file, or creating or modifying a module instance. The value used with this parameter is the file name and path to which a response file is read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the module instances in a domain:

```
$AdminTask manageItfimStsModuleInstance {-operation list
  -fimDomainName domain1}
```

Create a response file to create a new module instance that is based on a module type:

```
$AdminTask manageItfimStsModuleInstance {-operation createResponseFile
  -fimDomainName domain1 -moduleId moduleType1 -fileId c:\temp\modInst.rsp}
```

The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating a module instance or modifying an existing module instance. After you have created this file, open it with a text editor and define the attributes in it, so that they are correct for your environment.

Create a response file based on an existing module instance:

```
$AdminTask manageItfimStsModuleInstance {-operation createResponseFile
  -fimDomainName domain1 -uuid moduleInstance1 -fileId c:\temp\modInst.rsp}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating or modifying a module instance. After you have created this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a module instance:

```
$AdminTask manageItfimStsModuleInstance {-operation create
  -fimDomainName domain1 -fileId c:\temp\modInst.rsp}
```

Note: The file specified here is the response file and is used as input. Before running this command, open the response file with a text editor and ensure that the attributes defined in the file are correct for your environment.

Delete a module instance:

```
$AdminTask manageItfimStsModuleInstance {-operation delete
  -fimDomainName domain1 -uuid moduleInstance1}
```

View module instance details:

```
$AdminTask manageItfimStsModuleInstance {-operation view
  -fimDomainName domain1 -uuid moduleInstance1}
```

Modify a module instance

Run createResponseFile, as described in "Create a response file based on an existing module instance." The created file contains values for all the properties in the module instance. Edit this created file using a text editor. Save the file and then reload it into your environment using the modify command.

```
$AdminTask manageItfimStsModuleInstance {-operation modify
  -fimDomainName domain1 -uuid moduleInstance1
  -fileId c:\temp\modInst.rsp}
```

---

manageItfimStsChainMapping

Use the **manageItfimStsChainMapping** command to manage Security Trust Service (STS) Chain Mapping for IBM Tivoli Federated Identity Manager.

Purpose

The **manageItfimStsChainMapping** command manages the chain mapping identification associated with a trust chain. This command can perform the following operations to manage STS chain mapping when used with the appropriate parameters:

- list
- view
- delete
- create (using a response file)
- create the response file
- modify

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimStsChainMapping {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-uuid chain_mapping_uuid | chain_uuid
-fileId output_file | input_file
```

The use of these parameters depends on the operator you specify.

Parameters

The following parameters are available for use with the **manageItfimStsChainMapping** command:

-operation *operator*

Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 39. Values for the manageItfimSTSChainMapping -operation parameter

Value	Description and requirements
list	List all of the existing STS chain mappings.
view	View the details of a chain mapping. When you use this operator, you must also specify the **uuid** *chain_mapping_uuid* parameter.
delete	Delete a chain mapping. When you use this operator, you must also specify the **uuid** *chain_mapping_uuid* parameter.
createResponseFile	Create a response file to be used in creating a chain mapping. You can create a response file for a chain mapping based on an existing chain. Alternatively, you can create a response file based on an existing chain mapping. **New chain mapping based on an existing chain:** When you use this operator to create response file for a new chain mapping based on an existing chain, you must also specify the following parameters: **uuid** *chain_uuid* Specify the UUID of the chain based on which the chain mapping is to be created. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. **Based on existing chain mapping:** When you use this operator to create a response file that is based on an existing chain mapping, you must also specify the following parameters: **uuid** *chain_mapping_uuid* Specify the UUID of the existing chain mapping if you are creating a response file based on an existing chain mapping. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. After creating the response file, open it with a text editor and review the attributes defined in the file. Change the attributes as required by your environment. Then save and close the file. For information about the content of response files, see "Token module response files" on page 70.
create	Create a chain mapping using a response file. When you use this operator, you must also specify the following parameters: **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.

Table 39. Values for the manageItfimSTSChainMapping -operation parameter (continued)

Value	Description and requirements
delete	Delete a chain mapping. When you use this operator, you must also specify the **uuid** *chain_mapping_uuid* parameter provided with the UUID of the chain mapping to be deleted.
modify	Modify a chain mapping using a response file. When using this operator, you must specify the following parameters: **uuid** *chain_mapping_uuid* Specify the UUID of the chain mapping that you want to modify. **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-uuid *chain_mapping_uuid* | *chain_uuid*
An identifier string that uniquely identifies the resources on which to operate, either the UUID of the chain mapping or of the chain. The UUID of the chain is specified when creating a response file based on an existing chain.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file, or creating or modifying a chain mapping. The value used with this parameter is the file name and path for reading from a response file (input file) or writing to it (output file). The path and file name must be valid for the operating system being used.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the chain mappings in a domain:

```
$AdminTask manageItfimStsChainMapping {-operation list
  -fimDomainName domain1}
```

Create a response file to create a chain mapping based on an existing chain:

```
$AdminTask manageItfimStsChainMapping {-operation createResponseFile
  -fimDomainName domain1 -uuid chain1 -fileId c:\temp\fromChain.rsp}
```

Specified here is the name of the response file that you are creating with the command. Use this file as input for creating or modifying a chain mapping. After creating the file, open it with a text editor and define its attributes according to your environment.

Note: The uuid parameter contains the UUID of a chain and not a chain mapping.

Note: For creating an STS chain mapping that is not based on an existing chain mapping, use the console, as described in the Installation and Configuration Guide. To make minor modifications, run the command for creating a response file based on the existing chain mapping. Edit the resulting response file, and then run the command for creating the chain mapping.

Create a response file based on an existing chain mapping:

```
$AdminTask manageItfimStsChainMapping {-operation createResponseFile
  -fimDomainName domain1 -uuid chainMapping1
  -fileId c:\temp\chainMapping.rsp}
```

Note: In this case, the uuid parameter contains the UUID of a chain mapping and not a chain.

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating or modifying a chain mapping. After creating the file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create a chain mapping:

```
$AdminTask manageItfimStsChainMapping {-operation create
  -fimDomainName domain1 -fileId c:\temp\chainMapping.rsp}
```

Note: The file specified here is the response file and is used as input. Before running this command, open the response file with a text editor and ensure that the attributes defined in the file are correct for your environment.

Delete a chain mapping:

```
$AdminTask manageItfimStsChainMapping {-operation delete
  -fimDomainName domain1 -uuid chainMapping1}
```

View chain mapping details:

```
$AdminTask manageItfimStsChainMapping {-operation view
  -fimDomainName domain1 -uuid chainMapping1}
```

Modify a chain mapping

Run createResponseFile, as described in "Create a response file based on an existing chain mapping." The created file contains all the properties in the chain mapping. Edit the created file using a text editor. Save the file and then use the modify command to reload it into your environment.

```
$AdminTask manageItfimStsChainMapping {-operation modify
  -fimDomainName domain1 -fileId c:\temp\chainMapping.xml }
```

manageItfimStsChain

Use the **manageItfimStsChain** command to manage the IBM Tivoli Federated Identity Manager Security Trust Service (STS) chain.

Purpose

The **manageItfimStsChain** command, when used with the appropriate parameters, can perform the following operations to manage trust service chains:

- list
- view
- delete
- create (using a response file)

- create the response file

Syntax

The command syntax is as follows:

```
$AdminTask manageItfimStsChain {-operation operator -fimDomainName name
  [optional_parameters])
```

where the `-operation` parameter and its value *operator* and `-fimDomainName` and its value *name* are required. The optional parameters are:

```
-uuid chain_uuid
-fileId output_file | input_file
-instanceIds instance1 [,instance2,instance3,..]
-modes mode1 [,mode2,mode3,..]
```

The use of these parameters depends on the operator you specify.

Parameters

The following parameters are available for use with the **manageItfimStsChain** command:

-operation *operator*
Required parameter. The value used with this parameter specifies the operation to perform. Valid values are listed in the following table.

Table 40. Values for the manageItfimSTSChain -operation parameter

Value	Description and requirements
list	List all of the existing STS chains.
view	View the details of an STS chain. When you use this operator, you must also specify the **uuid** *chain_uuid* parameter.

Table 40. Values for the manageItfimSTSChain -operation parameter (continued)

Value	Description and requirements
createResponseFile	Create a response file to be used in creating an STS chain. You can create a response file for a new STS chain, or create a response file based on an existing STS chain. **New STS chain:** When you use this operator to create response file for a new STS chain, you must also specify the following parameters: **instanceIds** *instance1 [,instance2,instance3,..]* Specify the list of module instances that form the chain. Enter the module instance UUIDs as comma-separated values, in the appropriate order. **modes** *mode1 [,mode2,mode3,..]* Specify the modes for the module instances in the chain, corresponding to the values specified for the **-instanceIds** parameter. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. **Based on existing STS chain:** When you use this operator to create a response file that is based on an existing chain, you must also specify the following parameters: **uuid** *chain_uuid* Specify the uuid of the chain if you are creating a response file that is based on an existing STS chain. **fileId** *output_file* Specify the file name and path for the response file that is created by this command. After creating the response file, open it with a text editor and review the attributes that are defined in the file. Change the attributes accordingly, as required by your environment. Then save and close the file.
create	Create an STS chain using a response file. When you use this operator, you must also specify the following parameters: **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.
delete	Delete an STS chain. When you use this operator, you must also specify the **uuid** *chain_uuid* parameter.

Table 40. Values for the manageItfimSTSChain -operation parameter (continued)

Value	Description and requirements
modify	Modify a chain using a response file. Use this operation to add, remove, and reorder module instances of a chain. When you use this operator, you must also specify the following parameters: **uuid** *chain_uuid* Specify the uuid of the chain that you want to modify. **fileId** *input_file* Specify the file name and path for the response file that provides the input for this command. You can create the response file using the **createResponseFile** parameter.

-fimDomainName *name*
Required parameter. The value used with this parameter is the name of the domain on which the operation is to be performed. The name can be a string with characters of any type.

-uuid *name*
An identifier string that uniquely identifies the resource on which to operate.

-fileId *output_file* | *input_file*
This parameter is required if you are creating a response file, or creating or modifying a chain. The value used with this parameter is the file name and path to which a response file is read from (input file) or written to (output file). The path and file name must be valid for the operating system being used.

-InstanceIds *instance1 [,instance2,instance3,..]*
A comma-separated list of the module instances that are to be used for the chain assembly. Specify the UUIDs of the module instances in the proper order, separated by commas.

-modes
Comma-separated list of modes that the corresponding module instance ID operates in the chain. Specify the modes corresponding to the module instances in the chain.

Examples

The following examples show the correct syntax for several of the tasks that can be performed with this command:

List all the STS chains in a domain:

```
$AdminTask manageItfimStsChain {-operation list -fimDomainName domain1}
```

Create a response file to create a new STS chain:

In this case, create a response file for a username (validate) to SAML1.1 (issue) chain.

```
$AdminTask manageItfimStsChain {-operation createResponseFile
  -fimDomainName domain1 -instanceIds default-username,default-saml1_1
  -modes validate,issue -fileId c:\temp\utToSaml11.rsp}
```

The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating an STS chain or

modifying an STS chain. After creating this file, open it with a text editor and define the attributes in it, so that they are correct for your environment.

Note: For creating an STS chain that is not based on an existing chain, you can use the console, as described in the Installation and Configuration Guide. To make minor modifications, run the command for creating a response file based on the existing chain. Edit the resulting response file, and then run the command for creating the chain.

Create a response file based on an existing STS chain:

```
$AdminTask manageItfimStsChain {-operation createResponseFile
  -fimDomainName domain1 -uuid chain1_uuid idpsaml2
  -fileId c:\temp\chain1.rsp}
```

Note: The file specified here is the name of the response file that you are creating with the command. Use this file as input for creating an STS chain. After creating this file, open it with a text editor and ensure that the attributes defined in the file are correct for your environment.

Create an STS chain:

```
$AdminTask manageItfimStsChain {-operation create -fimDomainName domain1
  -fileId c:\temp\chain1.rsp}
```

Note: The file specified here is the response file and is used as input. Before running this command, open the response file with a text editor and ensure that the attributes defined in the file are correct for your environment.

Delete an STS chain:

```
$AdminTask manageItfimStsChain {-operation delete -fimDomainName domain1
  -uuid chain1_uuid}
```

View STS chain details:

```
$AdminTask manageItfimStsChain {-operation view -fimDomainName domain1
  -uuid chain1_uuid}
```

Modify an STS chain

First run createResponseFile, as described in "Create a response file based on an existing STS chain." The created file contains all of the properties in the chain. Edit the created file using a text editor. Save the file and reload it into your environment using the modify command.

Note: To add new module instances in the chain, add new elements in the response file specifying the order, mode, and uuid of the module instance. To delete a module instance, remove elements describing the uuid and mode for that module instance. To reorder, edit the Order numbers in the response file.

```
$AdminTask manageItfimStsChain {-operation modify -fimDomainName domain1
  -uuid chain1_uuid -fileId c:\temp\chainModify.xml}
```

Handling an unspecified name identifier

Learn how an unspecified name identifier is processed in a SAML 2.0 federation.

When a SAML 2.0 identity provider receives a single sign-on request, it typically contains a name identifier policy with a `Format` attribute specified by the service provider. The service provider indicates the name identifier format it wants to receive in the subject of an assertion from the identity provider. If the service provider sets the attribute to the value

urn:oasis:names:tc:SAML:1.1:nameidformat:unspecified, it is up to the identity provider to determine which name identifier format to use. The DefaultNameIDFormat configuration parameter of a federation or partner is used for this purpose.

The DefaultNameIDFormat parameter determines processing rules for the name identifier format when one of these conditions exists:

- if there is no explicit Format attribute included in the request
- if the Format attribute is set to urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

The value of the default name identifier format of the identity provider, if present, is obtained from the DefaultNameIDFormat parameter belonging to its corresponding partner configuration properties. Otherwise, it proceeds to retrieve the same parameter from the federation configuration properties. I

f the DefaultNameIDFormat parameter is not set at either partner or federation properties, it is obtained from the configuration parameter com.tivoli.am.fim.sts.saml.2.0.assertion.default.nameidformat that you set in the Default NameID Format for Assertion validation field, if present. If not, then the value defaults to urn:oasis:names:tc:SAML:2.0:nameid-format:persistent.

Note: You can specify the parameter com.tivoli.am.fim.sts.saml.2.0.assertion.default.nameidformat in the Default NameID Format for Assertion validation field of the Trust Service Chain Mapping Wizard.

The parameter treats the NameID included in the assertion as a string literal and no alias service lookup is used.

The DefaultNameIDFormat parameter can be configured to use one of the following permitted values:

- urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
- urn:oasis:names:tc:SAML:2.0:nameid-format:transient
- urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

The most common value is: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Each name identifier format works differently in processing single sign-on requests. For example, the persistent name identifier causes the server to use the alias service to look up or create an alias for the user of the federation and partner. The email address name identifier, however, causes the name identifier element to be populated with the user name of the currently authenticated user.

To use a different name identifier format other than the default value, configure the DefaultNameIDFormat parameter with a response file in the command-line interface. You can configure the parameter from the federation or partner level:

- "Configuring DefaultNameIDFormat (partner)" on page 313
- "Configuring DefaultNameIDFormat (federation level)" on page 314

Note: The DefaultNameIDFormat parameter from the partner configuration takes precedence over the property from the federation configuration.

Configuring DefaultNameIDFormat (partner)

Use the DefaultNameIDFormat parameter in the partner level to control the behavior of IBM Tivoli Federated Identity Manager when an unspecified format is used for name identifiers.

Before you begin

Ensure that you understand how to use the partner commands. See "manageItfimPartner" on page 247 for command details.

About this task

Configure the DefaultNameIDFormat parameter to define how an unspecified name identifier is processed in the partner level.

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, or Solaris

   ```
   wsadmin.sh
   ```

   Note: For more information about the options that can be specified when you run the wsadmin tool, see the WebSphere Application Server Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.
3. Save the partner configuration properties into a response file with the following command: `$AdminTask manageItfimPartner {-operation createResponseFile -fimDomainName name -federationName name -partnerName name -fileId /tmp/partner.out}`
4. Use the following commands to update the response file to include the new property with any of the permitted values:

   ```
   <void method="put">
   <string>DefaultNameIdFormat</string>
   <object class="java.util.ArrayList">
   <void method="add">
   <string>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</string>
   </void>
   </object>
   </void>
   ```

   The value used in this code is email address. See"Handling an unspecified name identifier" on page 311 for other permitted values.
5. Use the following commands to update the configuration in IBM Tivoli Federated Identity Manager: `$AdminTask manageItfimPartner {-operation modify -fimDomainName name -federationName name -partnerName name -fileId /tmp/partner.out}`
6. Use this command to reload the IBM Tivoli Federated Identity Manager configurations: `$AdminTask reloadItfimRuntime {-fimDomainName name}` You can also use the console to reload the IBM Tivoli Federated Identity Manager.

Configuring DefaultNameIDFormat (federation level)

Use the DefaultNameIDFormat parameter in the federation level to control the behavior of IBM Tivoli Federated Identity Manager when an unspecified format is used for name identifiers.

Before you begin

Ensure that you understand how to use the federation commands. See "manageItfimFederation" on page 211 for command details.

About this task

Configure the DefaultNameIDFormat parameter to define how an unspecified name identifier is processed in the federation level.

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, or Solaris

   ```
   wsadmin.sh
   ```

   Note: For more information about the options that can be specified when you run the wsadmin tool, see the http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.
3. Save the partner configuration properties into a response file with the following command: `$AdminTask manageItfimFederation {-operation createResponseFile -fimDomainName` *name* `-federationName` *name* `-fileId /tmp/fed.out}`
4. Use the following commands to update the response file to include the new property with any of the permitted values:

   ```
   <void method="put">
   <string>DefaultNameIdFormat</string>
   <object class="java.util.ArrayList">
   <void method="add">
   <string>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</string>
   </void>
   </object>
   </void>
   ```

   The value used in this code is email address. See"Handling an unspecified name identifier" on page 311 for other permitted values.
5. Use the following commands to update the configuration in IBM Tivoli Federated Identity Manager: `$AdminTask manageItfimFederation {-operation modify -fimDomainName` *name* `-federationName` *name* `-fileId /tmp/partner.out}`
6. Use this command to reload the IBM Tivoli Federated Identity Manager configurations: `$AdminTask reloadItfimRuntime {-fimDomainName` *name*`}` You can also use the console to reload the IBM Tivoli Federated Identity Manager.

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 give 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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 might 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 document 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 measurement 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 is for planning purposes only. The information herein is subject to change before the products described become available.

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. 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 IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

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 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.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

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.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

access token
In the context of OAuth, a string that represents authorization provided to the OAuth client. The string represents scopes and durations of access. It is granted by the resource owner and enforced by the OAuth or Authorization server.

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.

authorization code
In the context of OAuth, a code that the Authorization server generates when the resource owner authorizes a request.

authorization grant
In the context of OAuth, a grant that represents the resource owner authorization to access its protected resources. OAuth clients use an authorization grant to obtain an access token. There are four authorization grant types: authorization code, implicit, resource owner password credentials, and client credentials.

authorization server
A server that processes authorization and authentications.

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 the use of a self-posting form to establish and use a trusted session between two partners in a federation. Contrast with *browser artifact*.

certificate
In computer security, a digital document that binds a public key to the identity of the certificate owner. This digital document enables the certificate owner to be authenticated. A certificate is issued by a certificate authority and is digitally signed by that authority.

client A software program or computer that requests services from a server.

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). This technical standard 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.

keystore
In security, a file or a hardware cryptographic card where identities and private keys are stored, for authentication and encryption purposes. Some keystores also contain trusted, or public, keys.

Metadata
Data that describes a particular piece of information, such as settings for a configuration.

OAuth client
A third-party application that wants access to the private resources of the resource owner. The OAuth client can make protected resource requests on behalf of the resource owner once the resource owner grants it authorization.

OAuth server
Also known as the **Authorization server** in OAuth 2.0. The server that gives OAuth clients scoped access to a protected resource on behalf of the resource owner. An authorization server can also be the resource server.

partner
In data communications, the remote application program or the remote computer.

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.

private key
In secure communication, an algorithmic pattern used to encrypt messages that only the corresponding public key can decrypt. The private key is also used to decrypt messages that were encrypted by the corresponding public key. The private key is kept on the user system and is protected by a password.

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.

public key
In secure communication, an algorithmic pattern used to decrypt messages that were encrypted by the corresponding private key. A public key is also used to encrypt messages that can be decrypted only by the corresponding private key. Users broadcast their public keys to everyone with whom they must exchange encrypted messages.

refresh token
In the context of OAuth, a string that is used to obtain a new access token when the current access token expires.

resource owner
In the context of OAuth, a type of user capable of authorizing access to a protected resource.

resource server
The server that hosts the protected resources. It can accept and respond to protected resource requests using access tokens. The resource server might be the same server as the authorization server.

response file
A file containing predefined values such as parameters and values used to control the actions of a component in a predetermined manner.

request
An item that initiates a workflow and the various activities of a workflow.

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 start services across the Internet.

SOAP back channel
Communications that take place directly between two SOAP endpoints.

SPNEGO
Simple and Protected GSS API Negotiation Mechanism

stanza A group of lines in a file that together have a common function or define a part of the system. Stanzas are separated by blank lines or colons, and each stanza has a name.

syntax The rules for the construction of a command or statement.

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, and SOAP is used to transfer the data. A 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

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Printed in USA

SC23-6191-02